Theory and Design of Integrated Optical Isolators and Broadband Couplers Using Fresnel Zone Plates

by

Brad Gilbert Cordova

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Author .....
Department of Electrical Engineering and Computer Science
May 24, 2013

Certified by .....
Michael R. Watts
Associate Professor
Thesis Supervisor

Accepted by .....
Professor Leslie A. Kolodziejski
Chair of the Committee on Graduate Students

Theory and Design of Integrated Optical Isolators and Broadband Couplers Using Fresnel Zone Plates

by

Brad Gilbert Cordova

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis is divided into two main sections: the first containing the analysis of the broadband vertical coupler, and the second involving the theory and design of the integrated optical isolators. In the first part we propose, theoretically investigate, and numerically demonstrate a compact (less than $10\mu m$) broadband (more than $300nm$) fiber-chip vertical coupler. The structure utilizes a Fresnel lens, or more advanced integrated optics, placed above a short, ridge and deep etched, vertical coupler in a Si waveguide. This optics is placed in order to match the radiating fields to the fiber mode. We use semivectorial simulations with a simple stochastic optimization to design a good integrated optics without cylindrical constrains. Three-dimensional Finite-Difference Time-Domain (FDTD) simulations reveal $\sim 50\%$ fiber coupling efficiency and a bandwidth of $200nm$. In the second part we propose, theoretically investigate, and numerically demonstrate six designs of integrated optical isolators. We first derive analytically the value of the off-diagonal gyrotropic permittivity tensor element, ϵ_g. We then use this value to calculate a non-reciprocal phase shift in a Manganese, and a N/P doped silicon waveguide using analytic, perturbation, and a novel mode numeric approach. Finally, using the obtained magnitudes of the nonreciprocal phase shifts six integrated optical isolator designs are proposed.

Thesis Supervisor: Michael R. Watts
Title: Associate Professor

Acknowledgments

I would first like to thank my wife Esther, she is the most amazing woman I've ever met and she constantly supports me and makes me grow in ways that I neglected before I met her. I'm looking forward to a long interesting life together.

Next I would like to thank Mike, you first made me passionate about photonics and inspired me with your passion for the subject and ability to come up with really cool ideas. You've also given me useful advice about life, that has proven to be invaluable, and for that I am truly thankful.

I would also like to thank Ami for always being there to help me, every single time I had a question about photonics he was always there, literally, to help me even if he was busy. He has a true gift in explaining things, and I truly believe none of the stuff I accomplished would have been half as good if it wasn't for him.

I would also like to thank Cheri for meticulously helping me check my designs and bouncing ideas off of, for Ehsan for also being there every time I had questions, I admire your seemingly infinite patience, for Sasha helping me in the lab, where I spent most of the time lost and confused, for Zhan having interesting conversations about physics and photonics, for my roommate Purna, we've grown to become really good friends and develop interesting world views through long talks in our apartment, and for the rest of the Photonic Microsystem's lab group for making the last two years here at MIT great.

Finally, I wouldn't be anything without my parents. Both because they birthed me, but also because they shaped who I am as a person more than anyone else. And they have always been there to support me, and have shown me immense love that has never faded even though I've been so far from home over the last six years, thank you.

Contents

List of Figures

List of Tables

Chapter 1

Compact Broadband Vertical Coupler

1.1 Introduction

Many vertical fiber-chip coupler designs have been proposed and demonstrated in the last decade. It can be generally noted that these couplers are based on a grid coupling directly between the fiber and the on chip waveguide. Therefore, a constraint is applied on this grid. It must have at least the size of the fiber mode ($\sim 10\mu m$) in order to support a similar field distribution and maximize the coupling efficiency. This dimension constraint, as we show here, fundamentally limits the coupler bandwidth to about $60nm$ FWHM, which is at the same order as the fabrication errors for the center wavelength.

The design proposed and numerically demonstrated in this thesis includes a vertical coupler, much smaller than the fiber mode dimensions, with integrated optics above it for coupling to a single mode optical fiber. Finite-Difference Time-Domain (FDTD) verifies this design to have a bandwidth of $300nm$. In the telecom C-band, such a broadband coupler enables on chip Wavelength Division Multiplexing (WDM). Moreover, the size of this coupler takes an area smaller than the fiber tip placed above it.

This work is the first to separate between the coupling of the waveguide mode

(waveguide or vertical coupler) and the coupling of the fiber mode (fiber coupler), and hence the first to enable additional design flexibility of the vertical fiber-chip coupler. We show here the design and numerical evaluation of a method to couple a short vertical coupler with a fiber mode. The relation between the emitter length and the bandwidth will also be discussed in this thesis.

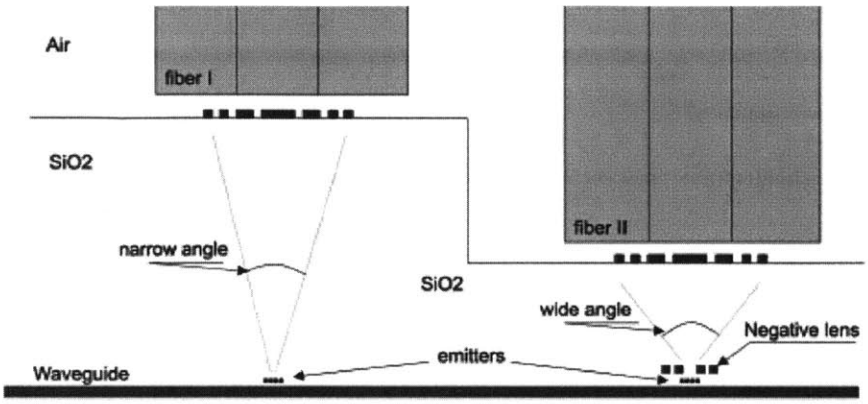


Figure 1-1: Two fiber couplers with (right) and without (left) the negative Fresnel lens for rapid expansion of the emitted field.

Our first concern, when designing a fiber coupler, is creating a mode, which is matched as closely possible to the fiber mode. To avoid the narrow band problem described above, and still be able to couple into a fiber mode, we propose here a split between the two problems. Step one, which is out of the scope of this thesis, is designing a short (low $L_{\#}$) vertical coupler and step two is coupling this vertical coupler to a fiber mode. In order to do so we simply apply some optics and enable propagation between the modes. Without loss of generality we can look at a waveguide to fiber coupling. By letting the short emitter field expand in the right amount of propagation, and then correct its phase using integrated Fresnel lens Figure 1-1a, we can practically match in a good agreement every short emitter field distribution to that of a fiber mode.

Further improvement to this design can be made using negative Fresnel lens right above the vertical coupler to cause the field shape to expand in a wider angle, achieving the fiber mode shape at a lower height. Figure 1-1b shows how this enables shorter distance to the original top lens plus fiber tip and eliminates the need of a thick oxide

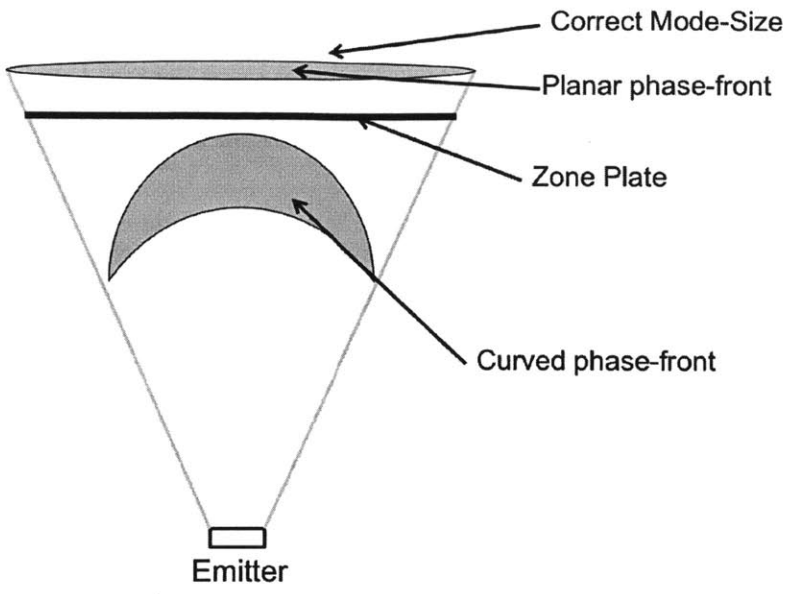


Figure 1-2: Diagrammatic description of Zone Plate operation.

layer above the waveguide coupler.

To design the simplest version of the figure above, after choosing a waveguide vertical coupler we run a 3D FDTD simulation of the waveguide and emitter alone looking only on the amplitude distribution of the emitted light at different heights above the waveguide and emitter. Finding the height which gives the amplitude distribution best fit to the fiber mode field distribution gives us the right height to place the phase correcting lens. This lens role is to collimate the light in order to make it similar to the fiber mode, and it is being constructed using Fresnel lens. First curve in Fig 3 shows the power coupling efficiency of 86.3% the fiber with lens and fiber tip height of $6.1\mu m$ above the waveguide and vertical coupler.

1.2 Vertical Coupler

For the purpose of analyzing a general vertical coupler, we consider a waveguide feeding a perturbation region with length L, and we monitor the wave emitting from that perturbation in a given direction with respect to the surface normal. It can be assumed, for simplicity, that the azimuth angle for the maximum coupling is set for

emission in the plane containing the waveguide direction and the chip surface normal. The emission is usually set to the first order refraction of the perturbation, so if the perturbation is designed for wavelength λ emitting in some angle $\theta_{\max}$, the phase of emission from lattice point l in the perturbation will be

$$\phi(x_l) = 2\pi \left(ml + \frac{x_l \sin(\theta_{\max})}{\lambda_0} \right), \tag{1.1}$$

where m is the refraction order, x_l is the perturbation period position. For general wavelength the phase is multiplied by so that it can be written as

$$\phi(x_l, \lambda) = \frac{2\pi}{\lambda} \left(ml\lambda_0 + x_l \sin(\theta_{\max}) \right) \tag{1.2}$$

and the far field in the $\theta_{\max}$ direction is then proportional to the sum

$$E_{FF} \propto \sum_l A(x_l) \exp\left(j(\phi(x_l, \lambda) - \frac{2\pi x_n \sin(\theta_{\max})}{\lambda}) \right) \tag{1.3}$$

$$= \sum_{l=1}^{L_\#} A(x_l) exp\left(\frac{j2\pi ml\lambda_0}{\lambda} \right). \tag{1.4}$$

$L_\#$ is the number of periods in the perturbation and $A(x_l)$ is the amplitude of radiating field in the perturbations period l. Assuming for simplicity that $A(x_l)$ is constant (same for all $x_l s$), and $m = 1$, we compare the normalized intensity to $\frac{1}{2}$, and numerically find the Full Width Half Maximum (FWHM) to equal.

Hence, for a common vertical coupler built with 20 to 25 periods with a central wavelength $1550nm$, the FWHM is inherently limited to $60nm$. A shorter coupler of four periods will enable FWHM$> 300nm$.

The vertical emitter size is $3\mu m \times 4.7\mu m$, with a grating period of $522nm$.

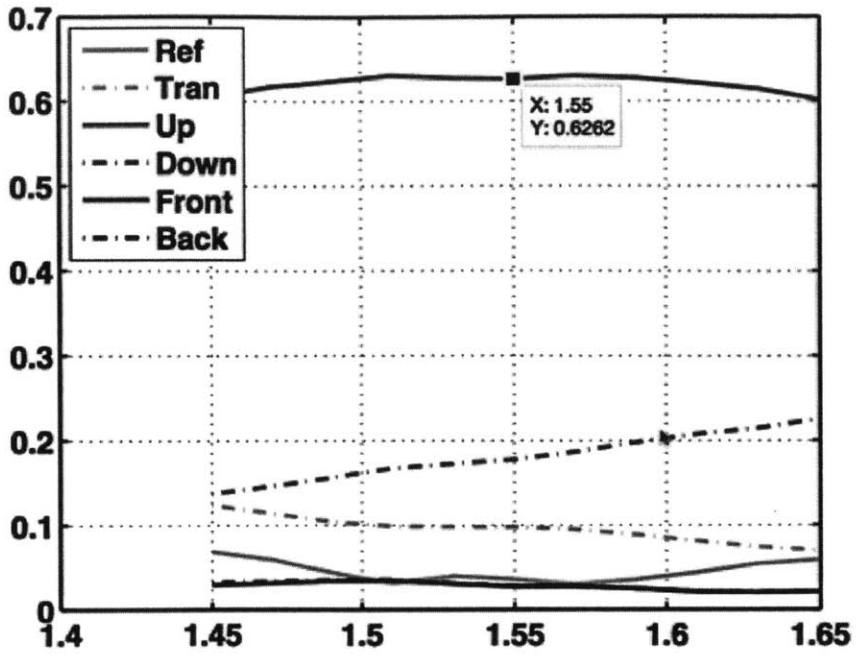


Figure 1-3: Coupling Efficiency of Emitter

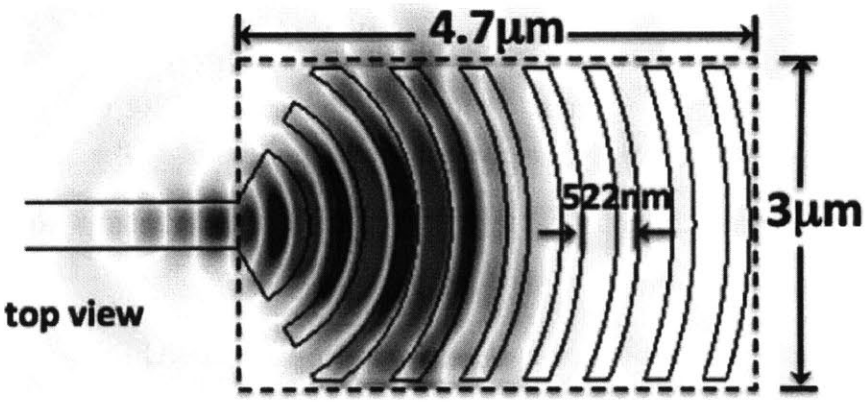


Figure 1-4: Emitter Design

1.3 Fresnel Zone Plate

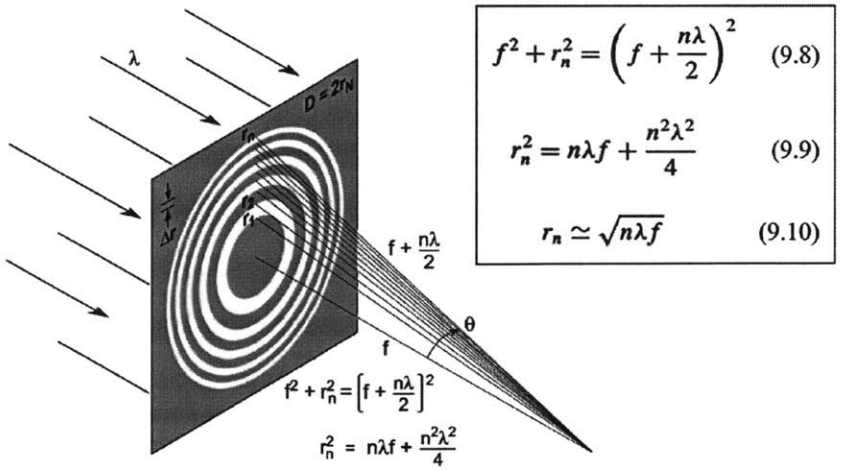
Now we design the diverging lenses. As described above, these lenses allow for a shorter device length compared to their absence because they accelerate the natural divergence of the emitted mode. First, we approached this problem using analytic methods to obtain a baseline solution to work off of, then created a more accurate solution using numeric methods as described below.

1.3.1 Analytic Design

To design the diverging lens proposed here we first assume the emitter to be a point source emitter, for calculation simplicity. Next we define a target height of cladding above the waveguide and first lens. In addition, we assume a collimated beam at the output of the waveguide vertical coupler with a typical diameter d corresponding to the vertical coupler length and width. The negative focal length of the bottom Fresnel lens will then be

$$f_1 \approx -\frac{h}{D/d - 1}, \tag{1.5}$$

where h is the second lens desired height above the first one and D is the fiber mode diameter. The second lens focal length is then $f_2 = h + f_1$. If the vertical coupler length and width are not the same, we can add astigmatism to the first lens and calculate $f_{1\leftrightarrow}$ and $f_{1\uparrow}$ using the two dimensions $d_{\leftrightarrow}$ and $d_{\uparrow}$ separately. The lens will then have elliptical line curves when top viewed. We assume in this part a near-field approximation for the waveguide coupler to first lens and for the second lens to fiber tip. Although the assumptions given here are rough, f_1 and f_2 can be refined for better overlap to the fiber mode.



$$f^2 + r_n^2 = \left(f + \frac{n\lambda}{2}\right)^2 \tag{9.8}$$

$$r_n^2 = n\lambda f + \frac{n^2\lambda^2}{4} \tag{9.9}$$

$$r_n \simeq \sqrt{n\lambda f} \tag{9.10}$$

Figure 1-5: Analytical diverging Fresnel Lens design

1.3.2 Numerical Design

Now that we have a baseline analytical design we can now optimize this design using numerical methods. This will be implemented by optimizing this analytical design by using stochastic optimization methods. This optimization step uses semi-vectorial simulation methods for speed enhancement. We then run the resultant lens mask through a low pass filter to avoid small points and sharp corners, and test it with a full vectorial FDTD simulation for verification.

In addition, by applying stochastic optimization when designing the lower lens, we can achieve better amplitude shape at the top lens corresponding to the fiber mode shape. In fact, it is not limited to any circular symmetry, and can take any deposited 2D shape of high indexed material in the lens plane.

Figure 1-6 shows the fabricated design variations, in order to compare simulation vs measurement for each step in the design processes.

Design #	Lens_11	Lens_12	Lens_2	Simulation
0	-	-	-	-
I	F	-	F'	C
II	F	F	F'	C
III	-	-	F'	C
IV	GAO	-	F'	SV
V	GAO	GAO	F'	SV+FDTD
VI	GAO	GAO'	F'	SV
VII	Use VI	Use VI	PC	SV+FDTD

Figure 1-6: Fabricated Design Variations. F stands for analytic Fresnel Lens design. GAO stands for Genetic Algorithm Optimization. SV stands for semi-vectorial simulation, and FDTD stands for Finite Difference Time Domain simulation.

Fresnel Propagation Integral

The semi-vectorial Fresnel Propagation integral is used to speed up computational efficiency, show in Figure 1-7.

In the ideal case we would only use FDTD to calculate coupling efficiency, but

$$U(x,y,z) = \frac{z}{i\lambda}\int\int_{\Sigma} U(\xi,\eta)\frac{e^{ikr_{01}}}{r_{01}^2}d\xi d\eta \qquad \because r_{10} = z\sqrt{1+\frac{(x-\xi)^2}{z^2}+\frac{(y-\eta)^2}{z^2}}$$

Figure 1-7: Semi-vectorial Frensnel Propagation integral and Phase Transformation scheme.

the computational efficiency of today's computers does not allow this to be a viable option. And since the propagation of the mode once it leave the vertical coupler is paraxial, this approximation is a reasonable substitution.

As shown in Figure 1-8, the simulation is carried out in a series of steps. Firstly, the amplitude and phase from a FDTD flux monitor placed directly above the vertical emitter is used as a starting field for the Fresnel Propagation Integral. Then using the integral we propagate it to the first diverging lens a distance L_1. Next we apply a phase shift to the propagated field, directly proportional to the dielectric arrangement of the first diverging lens. This process is carried out until we reach the optical fiber, wherein the field is saved. The outputted field is then used in a mode-overlap integral with the optical fiber mode to determine it's coupling efficiency, which is later used as the optimization parameter in the genetic algorithm.

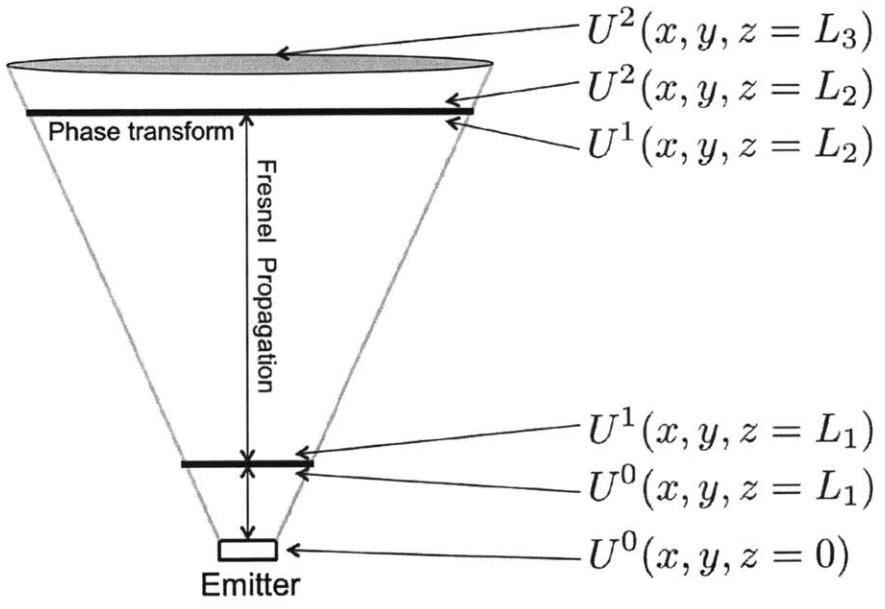


Figure 1-8: Mathematical representation of the Fresnel Propagation Integral.

Genetic Algorithm Optimization

Now that we have a computationally efficient algorithm for simulating the emission and coupling of the optical mode, we can now iterate this process to obtain local coupling maximums between the emitted mode and the fiber mode. Each diverging lens was created using a series of, $10nm$ spread randomly across each lens' area trial pixels, will a minimum of $10nm$ feature size due to fabrication specifications. If the addition contributed to higher mode coupling then the change was accepted and subsequent pixels surrounding the chosen pixel is also tested. The genetic algorithm is shown in Figure 1-9.

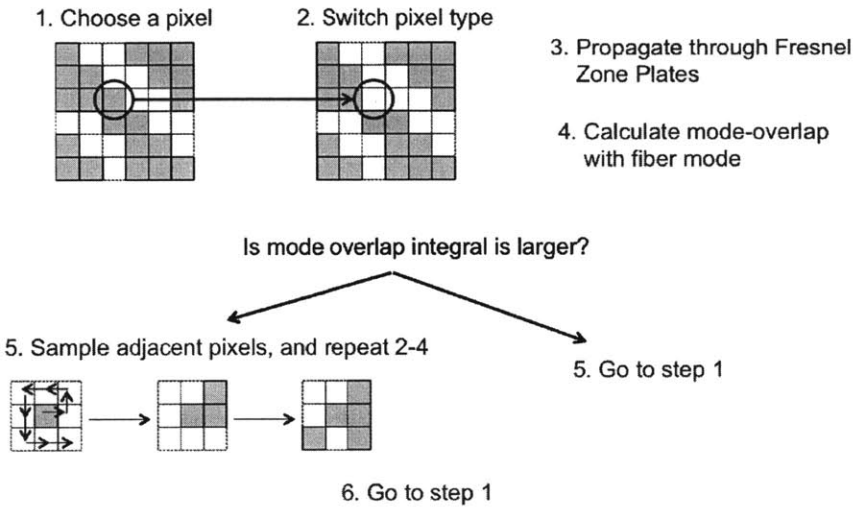


Figure 1-9: Diagrammatic representation of the genetic algorithm.

After each generation the amplitude coupling efficiency either improves or stays the same. After running this overnight on a large multiprocessor machine, we can obtain a solution which corresponds to a local coupling maximum. After running this with different starting conditions, then simulating each local solution using FDTD we can obtain an accurate approximation to the global solution.

In Figure 1-10 we see such a solution which corresponds to changing both the first and second diverging lens in identically.

Figure 1-11 is also a solution where the first and second diverging lenses are optimized independently, allowing for an even greater coupling efficiency. This design

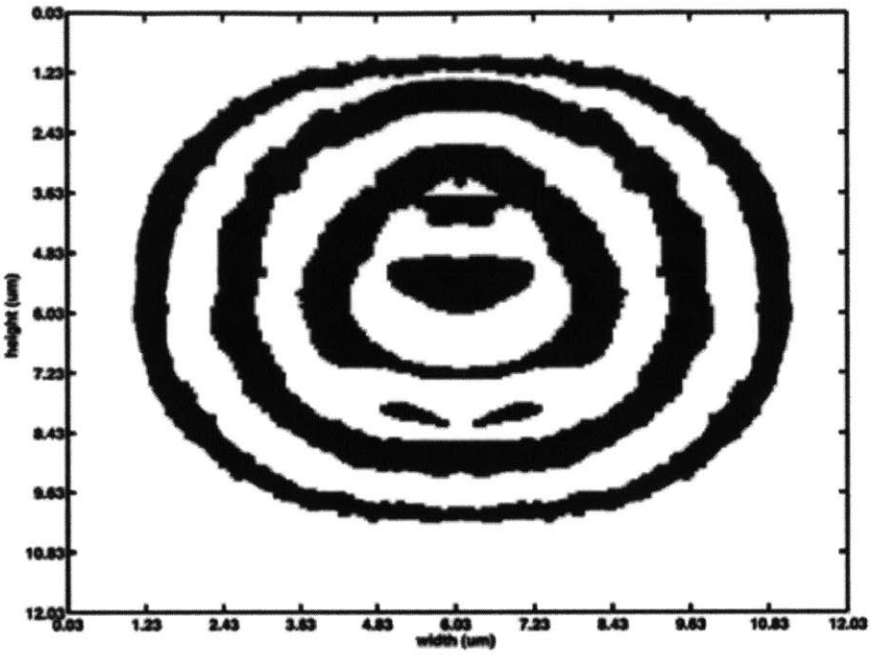


Figure 1-10: Local solution of diverging lens, where the first and second lens are identically optimized.

turns out to yield the highest coupling efficiency and is used as the result in this thesis.

In Figure 1-12 we see a comparison between the mode amplitude of a optical fiber mode, compared to that of the emitted mode.

Binary Phase Correction

Now that we have a mode which the amplitudes are now of the same size, the last step- as described above- is to correct the phase of the emitted mode, a curved phase front, so that it's optimized to match that of the fiber mode, a flat phase front.

For this a binary compensation algorithm is used. In the top most dielectric we can either decide for a $400nm$ Silicon Nitride dielectric to be present or not. And so we have developed a method for determining this. In the ideal case you would add

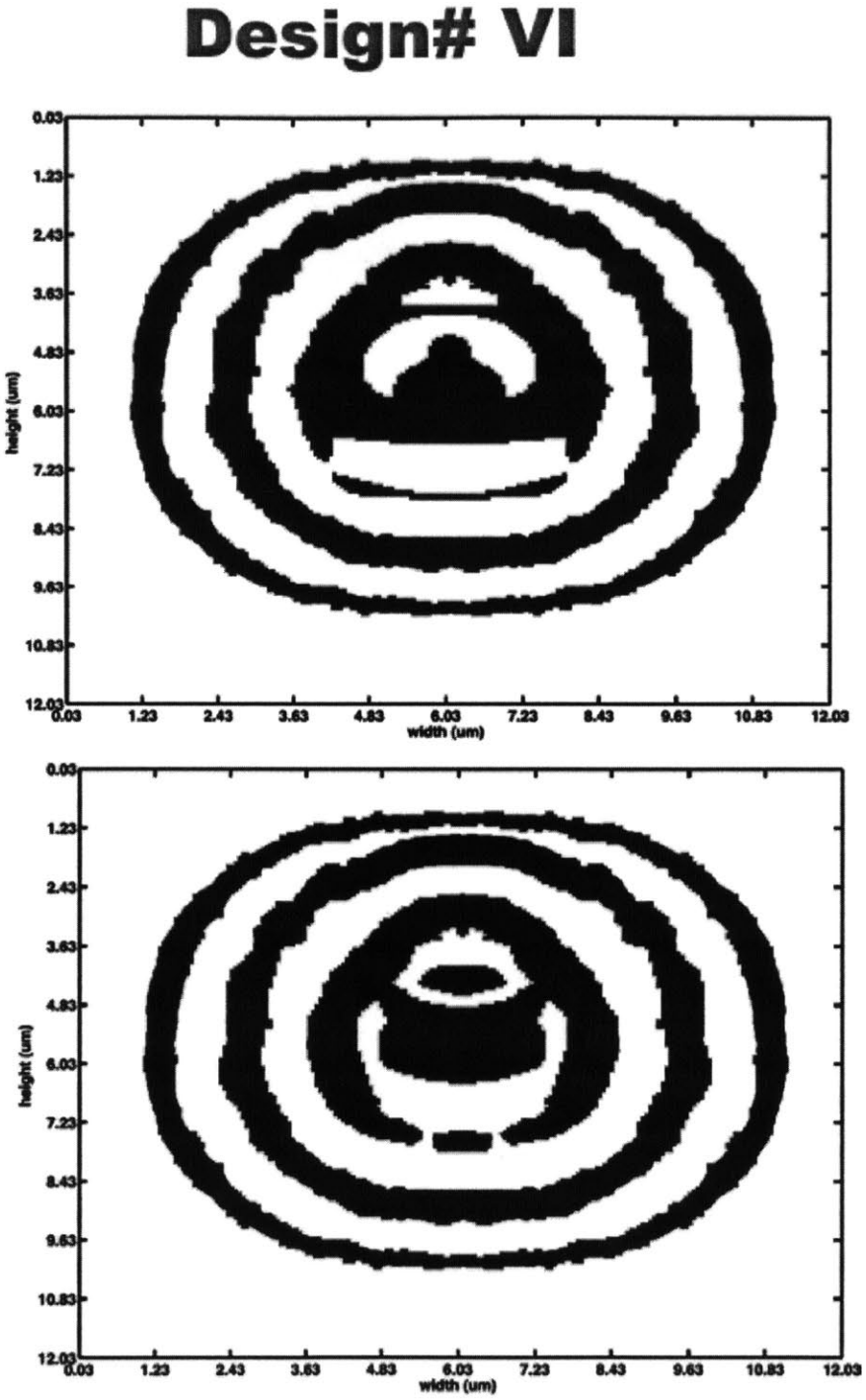


Figure 1-11: Local solution of diverging lens, where the first and second lens are independently optimized. This is the solution that yield the greatest coupling efficiency.

just the right amount of dielectric to phase shift the most "out of phase" parts of the mode so that they exactly match the parts of the mode "in phase". But we cannot

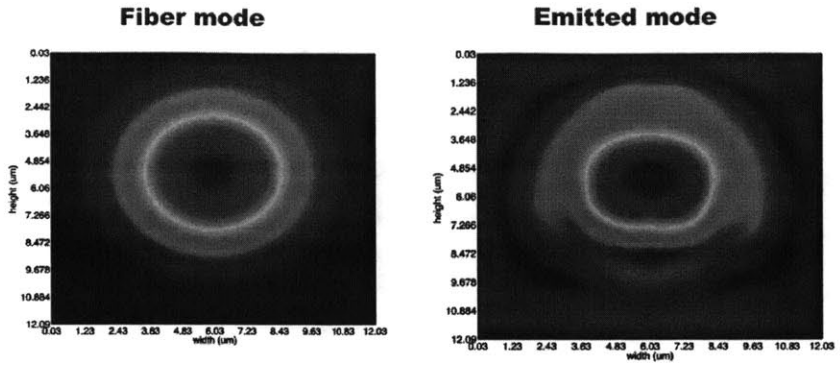


Figure 1-12: Comparison of the mode amplitudes of a fiber mode and the mode emitted from Design #VI

do this, because the dielectric cannot be applied in an analogue fashion, but rather in a digital fashion.

Thus, as shown in Figure 1-13 we first plot the phase distribution of the emitted wave on a polar plot. We then find the global maximum, weighted by it's standard deviation (fit with a Gaussian), and define an axis based on that maximum (shown as the solid red axis). Next choose the amount of desired phase shift we need for our application, α, and create another axis (shown as the dashed blue line) that is rotated by an angle of $\alpha/2$ from the original (solid red) axis. Then any phase that is on the opposite side of the dashed blue line gets phase shifted, that is at every location that corresponds to a phase that is on the opposite side of the second axis goes through the dielectric material located at the third lens, where as the locations of the mode that is on the right side of the second axis pass through the parts of the third lens where the dielectric is absent.

In Figure 1-14 we see the result of this phase correction. The image on the left is the uncorrected phase that emerges form the emitter and diverging lenses. The image on the right corresponds to the corrected phase that passes through the third dielectric lens. Note that the color scheme for the z-axis on both images is the same, we see that the image on the right is but a small domain of the uncorrected phase.

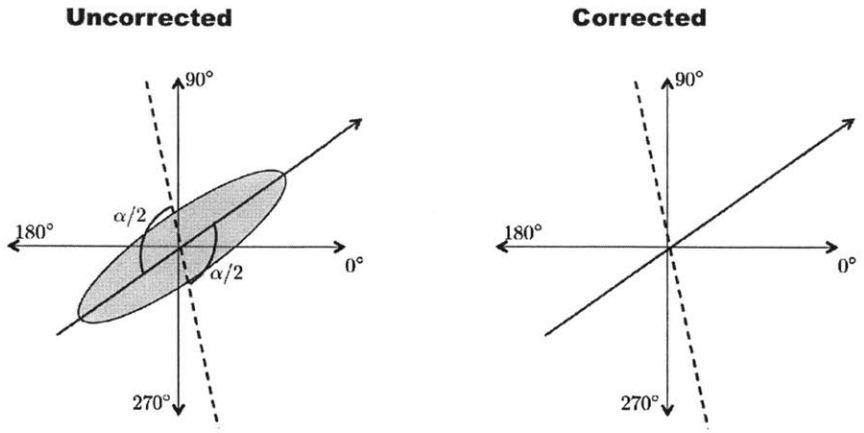


Figure 1-13: Graphical representation of the Binary Phase Correction algorithm.

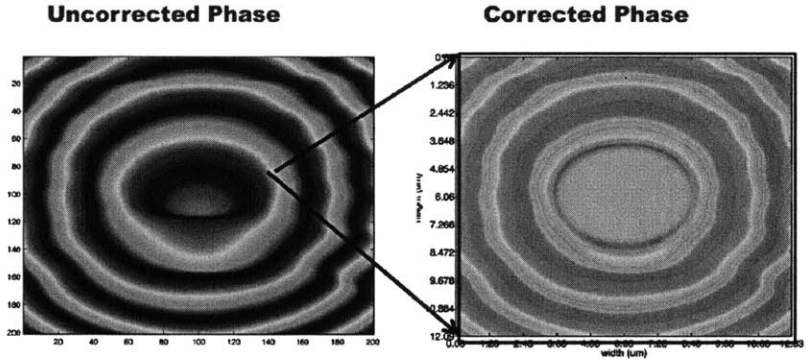


Figure 1-14: Comparison of the uncorrected (does not pass through third lens) vs. the corrected phase (passed through third lens)

1.3.3 FDTD Simulation

Finally, after we've run the stochastic optimization as a semivectorial simulation for the large number of iterations involved in it, we verified it in a FDTD simulation to prove its efficiency. Figure 1-15 shows the coupling efficiency for design $\#VI$.

1.4 Conclusions

In conclusion, we present a fiber-chip vertical coupler design that, by separating the waveguide coupling and the fiber coupling, enables broadband and small footprint. Using three-dimensional FDTD simulations, we show the coupling efficiency of the

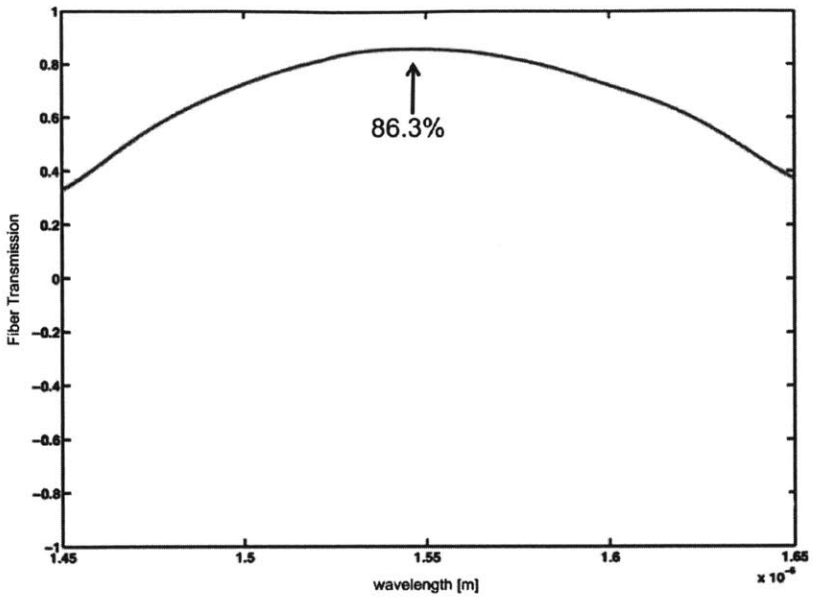


Figure 1-15: Bandwidth normalized to the power up coupled from the emitter

fiber coupler to reach 86.3% and an overall coupling bandwidth of $200nm$, with a total efficiency of 53%. Furthermore, the proposed structure uses realistic dimensions and commonly available materials. While efficient fiber-chip vertical couplers have been demonstrated, to our knowledge this is the first proposed and numerically verified compact coupler with a bandwidth of $200nm$ and the first proposed structure of its kind.

Chapter 2

Integrated Optical Isolators

2.1 Introduction

Microphotonics is surpassing electronics in terms of bandwidth distance product and power consumption for telecom and datacom applications. As the bandwidth increases photonics will replace electronics.

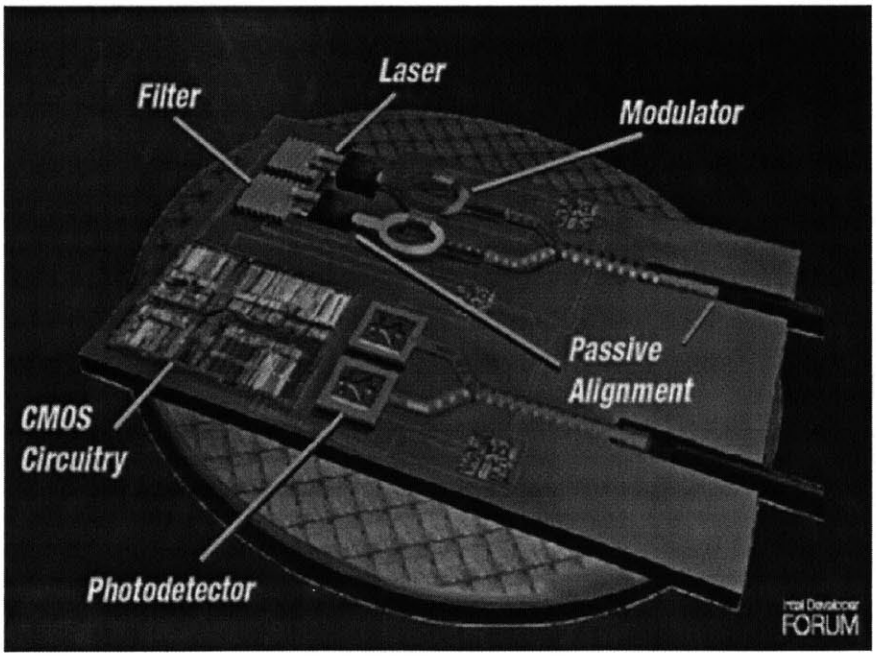


Figure 2-1: Diagram shows an optical computing device presented at Intel Developer Forum

There is one problem however, photonics has one key feature missing: the isolator- the photonics equivalent of the diode in electronics. Isolators are important for many optical oscillation phenomena, for example in a laser, back reflection into the cavity corrupts the temporal amplitude. Back reflection also distorts spectral characteristics of integrated systems.

So called, bulk isolators are common devices used in labs across the world. Bulk isolators work by taking advantage of the non-reciprocal nature of the Faraday effect. One could harness the Faraday rotation to create integrated optical isolators, however high index contrast silicon waveguides are highly birefringent and thus in general cannot support these circularly polarized waves.

This work tries to address and solve the problem of creating an integrated isolator, by proposing various designs and theoretical methods. The main theme will be to use the so called nonreciprocal phase shift, between forward and backward traveling modes in a Manganese doped Silicon waveguide. This allows us to create an isolator which does not suffer from the problems that the previously designed integrated Faraday rotation isolators suffer from.

2.2 Theory

When a Silicon waveguide is doped with Manganese impurities, the waveguide becomes intrinsic magneto-optic material (rather than associated with the presence of ferromagnetic clusters or a foreign ferromagnetic phase). When a Magnetic field field is applied along the transverse ($\hat{z}$) direction of the waveguide, the doped Silicon waveguide becomes gyrotropic in nature, with it's permittivity matrix matrix given by

$$\hat{\epsilon} = \begin{bmatrix} \epsilon & i\epsilon_g & 0 \\ -i\epsilon_g & \epsilon & 0 \\ 0 & 0 & \epsilon_z \end{bmatrix} \tag{2.1}$$

as will be derived in the next section. These $\epsilon_{x,y}$ and $\epsilon_{y,x}$ elements couple the

E_x and E_y fields which leads to the Faraday rotation phenomenon in the transverse plane. When you, however, apply an external magnetic field in the transverse plane ($\hat{x}$ or $\hat{y}$) you get a permittivity tensor of one of the following forms:

$$\hat{\epsilon} = \begin{bmatrix} \epsilon & 0 & i\epsilon_g \\ 0 & \epsilon & 0 \\ -i\epsilon_g & 0 & \epsilon_z \end{bmatrix} \tag{2.2}$$

$$\hat{\epsilon} = \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & \epsilon & i\epsilon_g \\ 0 & -i\epsilon_g & \epsilon_z \end{bmatrix} \tag{2.3}$$

This is the nature of the so called nonreciprocal phase shift, which couples the E_x and E_z or the E_y and E_z fields as opposed to the E_x and E_y fields as in the case of the Faraday rotation. This phenomenon is summarized in Figure 2-2.

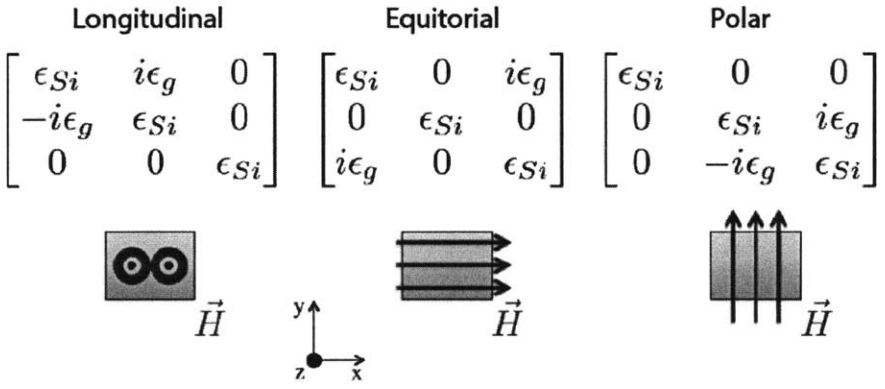


Figure 2-2: Matrix representations corresponding to externally applied magnetic field direction.

2.3 ϵ_g Calculation

For the design of integrated optical isolators it is first important to get an accurate estimate of the off-diagonal elements of the material's permittivity tensor. Without

the knowledge of this value, the rest of the calculations are meaningless. This number forms the basis of the rest of the designs.

In Manganese doped Silicon when an external magnetic field is applied the material becomes gyrotropic. When the DC magnetic field is applied along the transverse ($\hat{z}$) direction the permittivity tensor of the doped Silicon material takes the form

$$\hat{\epsilon} = \begin{bmatrix} \epsilon & i\epsilon_g & 0 \\ -i\epsilon_g & \epsilon & 0 \\ 0 & 0 & \epsilon_z \end{bmatrix} \tag{2.4}$$

That is using the Lorentz Force Law and Newton's second law, we can setup a force balance equation:

$$m\frac{d\mathbf{v}}{dt} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}). \tag{2.5}$$

Assuming time-harmonic excitation by the external magnetic field, and assuming the wave frequency is much greater than the electron collision frequency, the collision effect can be neglected and the free electron plasma can be considered to exist in a lossless medium. We then see that

$$-im\omega\mathbf{v} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B_0}) \tag{2.6}$$

Then,

$$-i\omega\mathbf{v} = \frac{q}{m}\mathbf{E} + \mathbf{v} \times \omega_\mathbf{c} \tag{2.7}$$

where $\omega_c = q\mathbf{B_0}/m$. Then multiplying Equation 2.7 by $\omega_\mathbf{c}$ we obtain that

$$-i\omega\mathbf{v} \cdot \omega_\mathbf{c} = \frac{q}{m}\mathbf{E} \cdot \omega_\mathbf{c} \tag{2.8}$$

Now we cross-multiply Equation 2.8 by $\omega_\mathbf{c}$ to obtain

$$-i\omega\mathbf{v}\times\omega_\mathbf{c} = \frac{q}{m}\mathbf{E}\times\omega_\mathbf{c}+(\mathbf{v}\times\omega_\mathbf{c})\times\omega_\mathbf{c} \tag{2.9}$$

$$= -\frac{q}{m}\omega_\mathbf{c}\times\mathbf{E}+\omega_\mathbf{c}(\mathbf{v}\cdot\omega_c)-\mathbf{v}\omega_c^2 \tag{2.10}$$

$$= -\frac{q}{m}\omega_\mathbf{c}\times\mathbf{E}+\frac{iq}{\omega m}\omega_\mathbf{c}(\omega_\mathbf{c}\cdot\mathbf{E})-\omega_c^2\mathbf{v} \tag{2.11}$$

Now substituting Equation 2.11 into Equation 2.7 we see that

$$\mathbf{J} = Nq\mathbf{v} \tag{2.12}$$

$$= \frac{Nq}{\omega^2-\omega_c^2}\left[-\frac{q}{m}\omega_\mathbf{c}\times\mathbf{E}+\frac{iq}{\omega m}\omega_\mathbf{c}(\omega_\mathbf{c}\cdot\mathbf{E})-\frac{i\omega q}{m}\mathbf{E}\right] \tag{2.13}$$

$$= \frac{-i\omega\epsilon_0\omega_p}{\omega^2-\omega_c^2}\left[-\frac{i}{\omega}\omega_\mathbf{c}\times\mathbf{E}+\frac{\omega_\mathbf{c}}{\omega^2}(\omega_\mathbf{c}\cdot\mathbf{E})-\mathbf{E}\right] \tag{2.14}$$

Then from Ampere's Law, we know that

$$\nabla\times\mathbf{H} = -i\omega\epsilon_0\mathbf{E}+\mathbf{J} \tag{2.15}$$

$$= -i\omega\left[i\epsilon_g\hat{z}\times\mathbf{E}+\epsilon E+\frac{\omega_p^2\omega_c^2\epsilon_0}{\omega^2(\omega^2-\omega_c^2)}\hat{z}\hat{z}\cdot\mathbf{E}\right] \tag{2.16}$$

$$= -i\omega\hat{\epsilon}\mathbf{E} \tag{2.17}$$

Thus we see that,

$$\epsilon_g = \epsilon_0\left[\frac{-\omega_p\omega_c}{\omega(\omega^2-\omega_c^2)}\right] \tag{2.18}$$

where $\omega_\mathbf{c} = q\mathbf{B_0}/m$, and $\omega_p = q\sqrt{\frac{N}{m_e\epsilon_0}}$. Thus since we know all of the constants on the L.H.S of Equation 2.18, we can obtain a value for ϵ_g based on the strength of B_0.

However, referring to the asymptote in Figure 2-3 a high ϵ_g near those asymptotes is unusable for real devices.

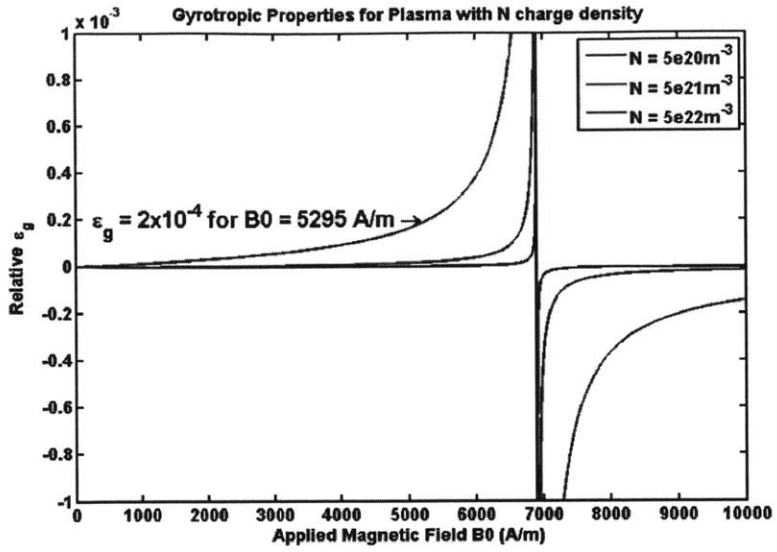


Figure 2-3: Resonance effect with applied magnetic field strength, B0

2.4 Non-Reciprocal Phase Shift Calculation

Now that we have a calculated number for ϵ_g we are now in place to calculate what our nonreciprocal phase shift will be. We will tackle this using three different methods of increasing accuracy to learn more about the qualitative and quantitative aspects of the non-reciprocal phase shift for design purposes.

2.4.1 Effective Index Calculation

We solve the exact Eigenvalue Equation for the Slab Waveguide

By solving the dispersion relation, one can obtain the propagation constant.

Each region of the Ridge Waveguide is treated as a Slab Waveguide, obtaining a 3D waveguide.

Start with Maxwells Eqs, with tensor permittivity

$$\nabla^i \times \underline{\mathbf{E}}^i = -j\omega\mu_0 \underline{\mathbf{H}}^i \tag{2.19}$$

$$\nabla^i \times \underline{\mathbf{H}}^i = -j\omega\hat{\epsilon}\underline{\mathbf{E}}^i \tag{2.20}$$

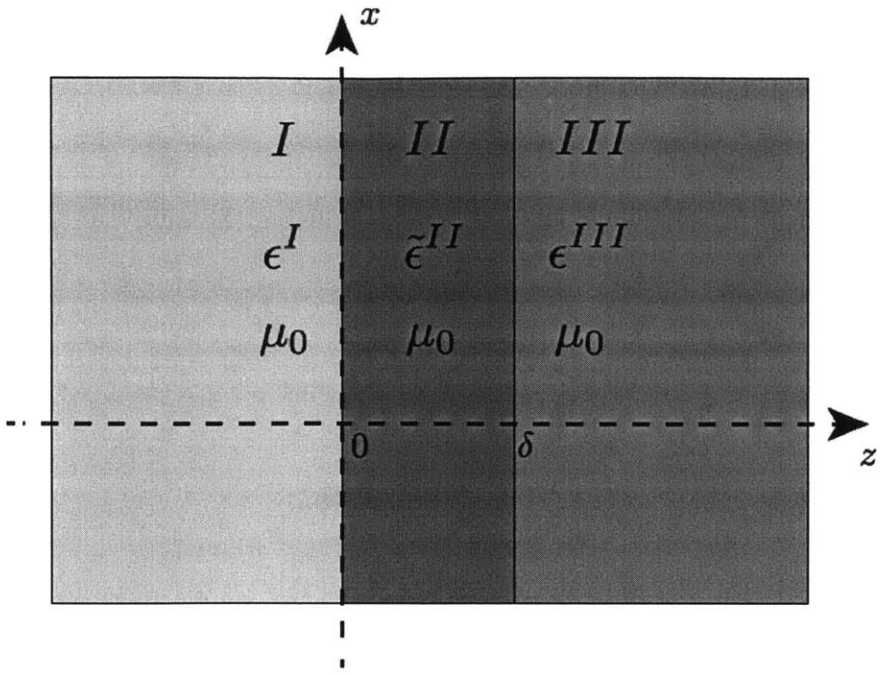


Figure 2-4: Dielectric configuration for the analytic calculation.

Solve for TM mode, by assuming

$$\underline{\mathbf{H_y}} = \underline{\mathbf{H_z}} = \underline{\mathbf{E_y}} = 0 \tag{2.21}$$

Derive vector wave eqn. by taking curl of Faradays Law, and inserting Amperes Law

$$\nabla(\nabla \cdot \underline{\mathbf{E}}^{II}) - \nabla^2 \underline{\mathbf{E}}^{II} = \omega^2 \mu_0 \hat{\epsilon}^{\mathbf{II}} \underline{\mathbf{E}}^{II} \tag{2.22}$$

Get dispersion relation, by assuming $exp(jk^{II}x)$ dependence

$$(k^{II})^2 = \omega^2 \mu_0 \epsilon_{eff} - \beta^2 \tag{2.23}$$

We then use trial wave functions of the form

Waveguide Width (μm)	$\delta\beta$ (cm^{-1})	Device Length (cm)
0.045	0.716	4.4
0.1	0.290	10.8
0.12	0.202	15.6

Table 2.1: Nonreciprocal phase shift calculations using the analytic calculation.

$$\underline{\mathbf{H_x}}^{I} = Ae^{K^I y}e^{-j\beta z} \tag{2.24}$$

$$\underline{\mathbf{H_x}}^{III} = Ae^{-K^{III} y}e^{-j\beta z} \tag{2.25}$$

$$\underline{\mathbf{H_x}}^{II} = [B\cos(k^{II}y) + C\sin(k^{II}y)]e^{-j\beta z} \tag{2.26}$$

Solve for E fields from Amperes Law, evoke boundary conditions to get the determinantal equation by which we can calculate our non-reciprocal phase shift.

$$0 = \tan(k^{II}\delta)\left[\frac{1}{\epsilon_{eff}^2}\left((k^{II})^2 - \frac{\beta^2}{\theta^2}\right) \pm j\frac{\beta}{\theta}\frac{1}{\epsilon_{eff}}\left(\frac{K^I}{\epsilon^I} - \frac{K^{III}}{\epsilon^{III}}\right) - \frac{K^1 K^{III}}{\epsilon^I \epsilon^{III}}\right] \tag{2.27}$$

$$-\frac{k^{II}}{e_{eff}}\left(\frac{K^I}{\epsilon^I} - \frac{K^{III}}{\epsilon^{III}}\right) \tag{2.28}$$

where

$$\theta = \frac{\epsilon_{Si}}{j\epsilon_g} \tag{2.29}$$

We see then that there are two solutions to Equation 2.27 for β traveling in opposite directions, which leads to a non-reciprocal phase shift.

2.4.2 Perturbation Method Calculation

First, we will make the standard assumption that the waveguide is homogeneous in the z-direction, and that the material is isotropic and lossless. The propagating modes, as before, can be described as

$$\left.\begin{matrix}\mathbf{E}(x,y)\\ \mathbf{H}(x,y)\end{matrix}\right\} e^{i(\beta z - \omega t)} \tag{2.30}$$

where $\mathbf{E}$ and $\mathbf{H}$ are the components of the mode amplitudeprofile, β denotes the

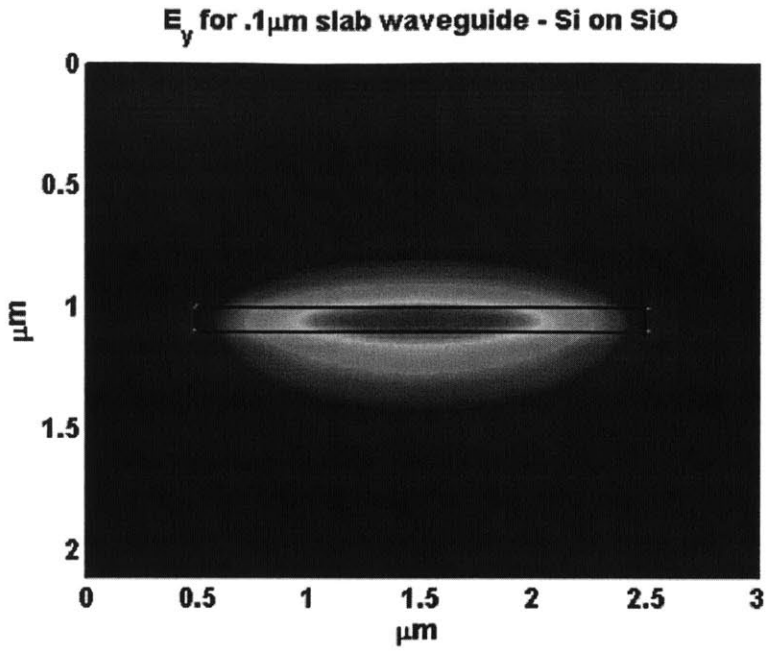


Figure 2-5: Choice of waveguide width is tradeoff between $\delta\beta$ and mode confinement

propagation constant, $\omega = ck_0$, is the angular frequency, and $k_0 = \frac{2\pi}{\lambda}$ is the vacuum wave number with λ denoting the vacuum wavelength. Maxwell's Equations yield the following differential equations for the transverse Electric and Magnetic fields

$$\begin{bmatrix} k_0^2\epsilon + \partial_x \frac{1}{\epsilon}\partial_x \epsilon + \partial_y^2 & \partial_x \frac{1}{\epsilon}\partial_y \epsilon - \partial_x \partial_y \\ \partial_y \frac{1}{\epsilon}\partial_x \epsilon - \partial_y \partial_x & k_0^2\epsilon + \partial_y \frac{1}{\epsilon}\partial_y \epsilon + \partial_x^2 \end{bmatrix} \begin{bmatrix} E_x \\ E_y \end{bmatrix} = \beta^2 \begin{bmatrix} E_x \\ E_y \end{bmatrix} \tag{2.31}$$

$$\begin{bmatrix} k_0^2\epsilon + \partial_y \frac{1}{\epsilon}\partial_y \epsilon + \partial_x^2 & -\partial_y \frac{1}{\epsilon}\partial_x \epsilon + \partial_x \partial_y \\ -\partial_x \frac{1}{\epsilon}\partial_y \epsilon + \partial_y \partial_x & k_0^2\epsilon + \partial_x \frac{1}{\epsilon}\partial_x \epsilon + \partial_y^2 \end{bmatrix} \begin{bmatrix} H_x \\ H_y \end{bmatrix} = \beta^2 \begin{bmatrix} H_x \\ H_y \end{bmatrix} \tag{2.32}$$

The longitudinal z components are then determined by Maxwell's divergence equations

$$\nabla \cdot \epsilon \mathbf{E} = 0 \tag{2.33}$$

$$\nabla \cdot \mu \mathbf{H} = 0 \tag{2.34}$$

where $\epsilon = n_0^2$ denotes the permittivity, n_0 is the isotropic refractive index, and μ is the permeability. At optical frequencies, $\mu \approx 1$. It turns out that Equations 2.31 and 2.32 are equivalent.

In fact, they completely describe the full vectorial modal fields $\mathbf{E}$ and $\mathbf{H}$. However, we usually see that one field component usually dominates and so if the non-dominant components are neglected, one can derive the simplified, so called, semi vectorial equations for the so called quasi-TE and quasi-TM modes:

$$\text{quasi-TE:} \quad \mathbf{E} = \begin{bmatrix} 0 \\ E_y \\ E_z \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} \tag{2.35}$$

corresponding to the wave-equation:

$$\left(k_0^2\epsilon + \partial_x^2 + \partial_y \frac{1}{\epsilon}\partial_y \epsilon\right) E_y = \beta_{TE}^2 E_y, \tag{2.36}$$

$$\text{quasi-TM:} \quad \mathbf{E} = \begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 0 \\ H_y \\ H_z \end{bmatrix} \tag{2.37}$$

corresponding to the wave-equation:

$$\left(k_0^2\epsilon + \partial_x^2 + \partial_y \frac{1}{\epsilon}\partial_y \epsilon\right) H_y = \beta_{TM}^2 H_y \tag{2.38}$$

For planar waveguides the partial derivatives with respect to y vanish, yielding analytically solvable equations.

As described in the introduction, the magneto-optical properties of the material are described by the non-diagonal components of the permittivity tensor:

$$\hat{\epsilon} = n_0^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + K \begin{bmatrix} 0 & M_z & -M_y \\ -M_z & 0 & M_x \\ M_y & -M_x & 0 \end{bmatrix} \tag{2.39}$$

$$= n_0^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \boldsymbol{\Delta}\hat{\epsilon} \tag{2.40}$$

Where $M_j, j = x, y, z$ denotes the vector components of the material's manetization. K is a complex material parameter $K = K' + jK''$, where K'' is related to the Faraday rotation by

$$\Theta_{F,sat} = -k_0 \frac{K'' M_s}{2n_0}. \tag{2.41}$$

The real part K' determines the Faraday ellipticity, which is neglected in this thesis.

However, as stated earlier because of the inherent birefringence of silicon waveguides, we will be harnessing the nonreciprocal phase shift effect instead of the Faraday rotation effect. Thus, we see that the elements of the gyrotropic part $\hat{\epsilon}$ of the permittivity tensor are small compared with n_0^2 and can thus be treated by perturbation theory. The component M_z which is parallel to the waveguide axis, gives rise to TE TM mode coupling and thus to mode conversion, which is similar to Faraday rotation in bulk media; this is discussed more in the Future Work section. The components M_x and M_y , which are perpendicular to that axis, induce a change $\delta\beta$ of the propagation constant β, which depends on the propagation direction, forward or backward:

$$\beta^{\text{forward}} = \beta + \delta\beta \tag{2.42}$$

$$\beta^{\text{backward}} = \beta - \delta\beta \tag{2.43}$$

resulting in a difference $\Delta\beta = 2\delta\beta$ between the forward and backward propagation constants; β denotes the unperturbed propagation constant. This nonreciprocal phase shift does not exist in the case of light propagation in bulk materials because the effect can be viewed as being immediately related to material discontinuities.

Similar to how this is treated in quantum-mechanics, perturbation theory for $\delta\beta$ yields

$$\delta\beta = \frac{\omega\epsilon_0}{N}\int\int \mathbf{E}^*\boldsymbol{\Delta}\hat{\epsilon}\mathbf{E}dxdy \tag{2.44}$$

normalized by the power flow in the z direction:

$$N = \left(\int\int \mathbf{E}\times\mathbf{H}^* + \mathbf{E}^*\times\mathbf{H}\right)dxdy. \tag{2.45}$$

We also see that the components M_x and M_y also cause TE-TM mode coupling that is usually negligible compared with the coupling produced by M_z. Now we can use the semi vectorial approximation for the quasi-TM and quasi-TE modes. Assume $\mathbf{M}$ is parallel to the x-axis, thus we see that

$$\mathbf{E}^*\boldsymbol{\Delta}\hat{\epsilon}\mathbf{E} = \mathbf{E}^*\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & i\epsilon_g \\ 0 & -i\epsilon_g & 0 \end{bmatrix}\begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix} \tag{2.46}$$

$$= \mathbf{E}^*\begin{bmatrix} 0 \\ i\epsilon_g E_y \\ -i\epsilon_g E_z \end{bmatrix} \tag{2.47}$$

$$= i\epsilon_g E_z E_y^* - i\epsilon_g E_z^* E_y \tag{2.48}$$

Now recall from our divergence relations Equation 2.33,

$$\nabla \cdot \mathbf{E} = 0 \tag{2.49}$$

$$\frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_z}{\partial z} + = 0 \tag{2.50}$$

Since for the quasi-TE mode $E_x = 0$ this implies

$$\frac{\partial E_y}{\partial y} = -\frac{\partial E_z}{\partial z} = \frac{\partial}{\partial z}\left(E \cdot e^{-i\beta z}\right) \tag{2.51}$$

$$= i\beta_{TE} E_z \tag{2.52}$$

It then follows that

$$E_z = \frac{-i}{\beta}\frac{\partial E_y}{\partial y} \tag{2.53}$$

Now plugging Equation 2.53 into 2.50 yeilds

$$\mathbf{E}^* \mathbf{\Delta} \hat{\epsilon} \mathbf{E} = i\epsilon_g E_y^* \frac{i}{\beta}\partial_y E_y - i\epsilon_g \frac{i}{\beta}\partial_y E_y \tag{2.54}$$

$$= \frac{2\epsilon_g}{\beta} E_y \partial_y E_y \tag{2.55}$$

Now finally substituting Equation 2.55 into Equation 2.44 yields,

$$\delta\beta = \frac{\omega\epsilon_0 \int\int \frac{2\epsilon_g}{\beta} E_y \partial_y E_y dx dy}{2N}, \tag{2.56}$$

simplifying we find that, the nonreciprocal phase shift for the TE mode takes the form of

$$\delta\beta^{TE} = \frac{\omega\epsilon_0}{\beta^{TE} N} \int\int \epsilon_g E_y \partial_y E_y dx dy. \tag{2.57}$$

Where, we see thatϵ_g present in the integral since it only multiplies that areas of the field that it interacts with. Using the same derivation as above we also see that if

M is parallel to the y-axis, the non-reciprocal phase shift for the TM mode is, where again we neglect the second order derivative with respect to x,

$$\delta\beta^{TM} = \frac{2\beta^{TM}}{\omega\epsilon_0 N}\int\int \frac{\epsilon_g}{n_0^4} H_y \partial_x H_y dxdy. \tag{2.58}$$

Thus, we see from our perturbative analysis that in order achieve a large nonreciprocal phase shift according to Equations 2.57 and 2.58, it is essential to create a strong discontinuity of the Faraday rotation at the maximum of the mode intensities or to position such a discontinuity at the mode intensity maximum, respectively. Equations 2.57 and 2.58 show that the magnitude and sign of the nonreciprocal phase shifts $\Delta\beta^{TE,TM}$ depend on the magnitude and sign of the Faraday rotation and on the geometry of the waveguide.

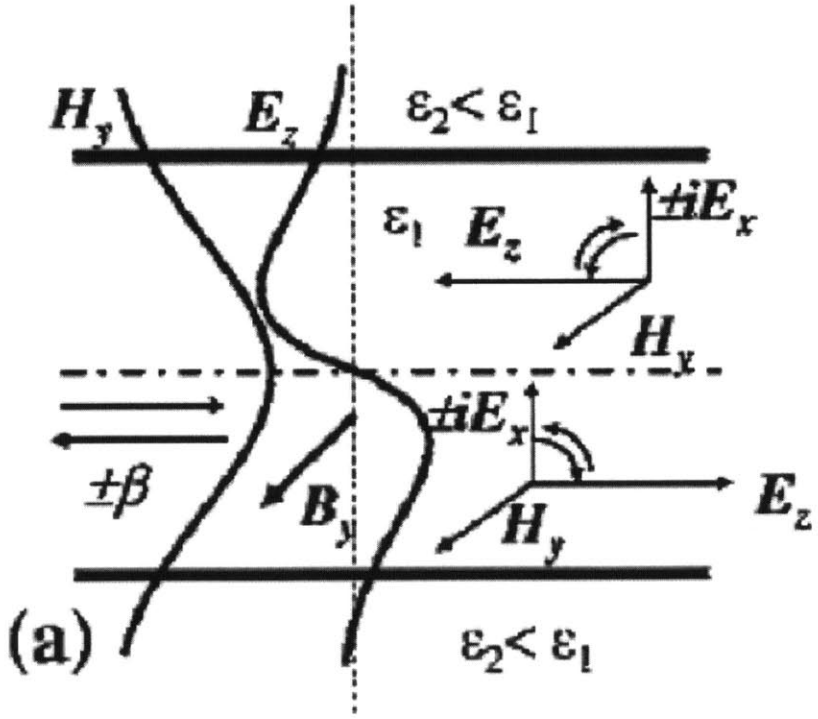


Figure 2-6: Longitudinal electric field component changes sign, and so anti-symmetry in the mode cancels phase shift. Image from [7].

Conclusively, we see the nonreciprocal manifests itself in the fact that in the TM wave the non-dominant longitudinal component of E is shifted by $\pm 90°$ from

Waveguide Width (μm)	$\delta\beta$ (cm^{-1})
0.045	0.854
0.1	0.332
0.12	0.250

Table 2.2: Nonreciprocal phase shift calculations using perturbation theory.

the z component of E depending on the direction of propagation. We can see this as an elliptical polarization of the mode in the propagation plane (as compared to the transverse plane, as is the case in Faraday rotation). Thus, if the waveguide material is magneto-optic, applied transverse magnetic field can cause phase shifts of opposite signs for forward and backward propagating waves. Unfortunately, the longitudinal component of electric field in the fundamental TM wave changes sign near the middle of the waveguide Figure 2-6 and in a uniform symmetric waveguide, complete cancelation of the nonreciprocal effect ensues. This is why we must use the

$$\begin{bmatrix} \epsilon_{xx} & 0 & i\epsilon_g \\ 0 & \epsilon_{yy} & 0 \\ -i\epsilon_g & 0 & \epsilon_{zz} \end{bmatrix}, \text{ or } \begin{bmatrix} \epsilon_{xx} & 0 & 0 \\ 0 & \epsilon_{yy} & i\epsilon_g \\ 0 & -i\epsilon_g & \epsilon_{zz} \end{bmatrix} \tag{2.59}$$

permittivity tensor components, which correspond to transverse magnetic fields, not longitudinal as in the case of the Faraday rotation.

2.4.3 Full anistropic Mode Solver Calculation

We start by writing Faraday's and Ampere's Laws in the frequency domain as

$$\nabla \times \underline{\mathbf{E}} = -j\omega\hat{\mathbf{u}}\underline{\mathbf{H}} \tag{2.60}$$

$$\nabla \times \underline{\mathbf{H}} = j\omega\underline{\mathbf{D}} \tag{2.61}$$

Where ω is the angular frequency of the harmonic fields, and $\bar{\mathbf{u}}$ represents a diagonal permeability tensor with diagonal elements μ_x, μ_y, and μ_z.

Thus we can write Equations 2.60 and 2.61 in terms of the following six component

equations, in terms of six electric and magnetic field components:

$$-j\omega\mu_x H_x = \frac{\partial E_z}{\partial y} + j\beta E_y \tag{2.62}$$

$$-j\omega\mu_y H_y = -j\beta E_x - \frac{\partial E_z}{\partial x} \tag{2.63}$$

$$-j\omega\mu_z H_z = \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \tag{2.64}$$

$$j\omega D_x = \frac{\partial H_z}{\partial y} + j\beta H_y \tag{2.65}$$

$$j\omega D_y = -j\beta H_x - \frac{\partial H_z}{\partial x} \tag{2.66}$$

$$j\omega D_z = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \tag{2.67}$$

$$\tag{2.68}$$

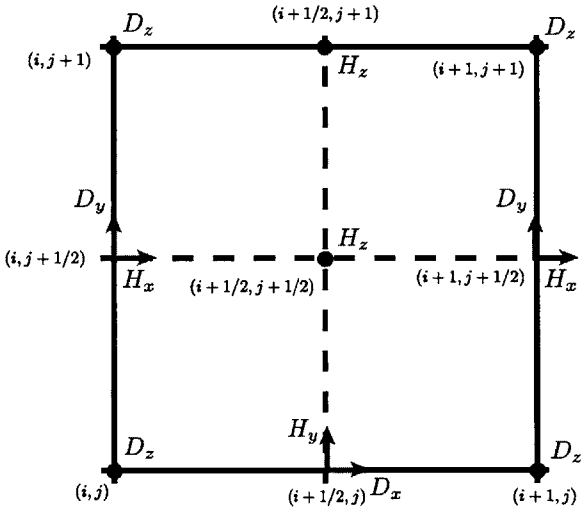


Figure 2-7: Yee Lattice

Therefore, we can map Equations 2.62 - 2.67 onto the Yee Lattice using Finite Difference discretization, yielding the corresponding Finite Difference equations:

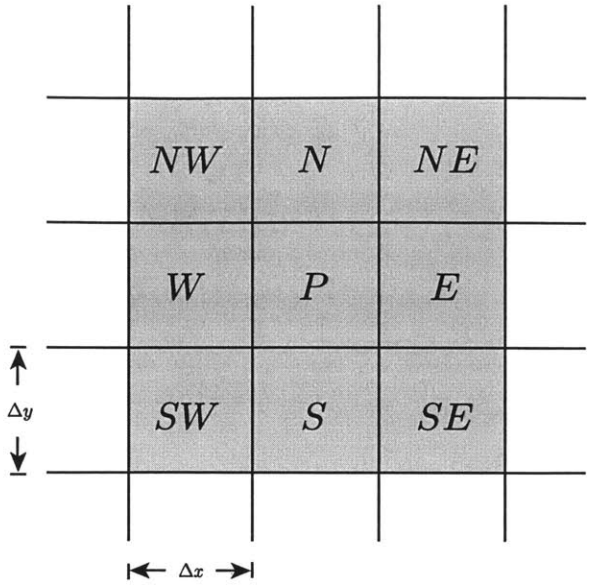


Figure 2-8: Labeling Scheme

$$-j\omega\mu^{x}_{(i,j+\frac{1}{2})}H^{x}_{(i,j+\frac{1}{2})} = \frac{E^{z}_{(i,j+1)} - E^{z}_{(i,j)}}{\Delta y} + j\beta E^{y}_{(i,j+\frac{1}{2})} \tag{2.69}$$

$$-j\omega\mu^{y}_{(i+\frac{1}{2},j)}H^{y}_{(i+\frac{1}{2},j)} = -j\beta E^{x}_{(i+\frac{1}{2},j)} - \frac{E^{z}_{(i+1,j)} - E^{z}_{(i,j)}}{\Delta x} \tag{2.70}$$

$$-j\omega\mu^{z}_{(i+\frac{1}{2},j+\frac{1}{2})}H^{z} = \frac{E^{y}_{(i+1,j+\frac{1}{2})} - E^{y}_{(i,j+\frac{1}{2})}}{\Delta x} - \frac{E^{x}_{(i+\frac{1}{2},j+1)} - E^{x}_{(i+\frac{1}{2},j)}}{\Delta y} \tag{2.71}$$

$$j\omega D^{x}_{(i+\frac{1}{2},j)} = \frac{H^{z}_{(i+\frac{1}{2},j+\frac{1}{2})} - H^{z}_{(i+\frac{1}{2},j-\frac{1}{2})}}{\Delta y} + j\beta H^{y}_{(i+\frac{1}{2},j)} \tag{2.72}$$

$$j\omega D^{y}_{(i,j+\frac{1}{2})} = -j\beta H^{x}_{(i,j+\frac{1}{2})} - \frac{H^{z}_{(i+\frac{1}{2},j+\frac{1}{2})} - H^{z}_{(i-\frac{1}{2},j+\frac{1}{2})}}{\Delta x} \tag{2.73}$$

$$j\omega D^{z}_{(i,j)} = \frac{H^{y}_{(i+\frac{1}{2},j)} - H^{y}_{(i-\frac{1}{2},j)}}{\Delta x} - \frac{H^{x}_{(i,j+\frac{1}{2})} - H^{x}_{(i,j-\frac{1}{2})}}{\Delta y} \tag{2.74}$$

This, can be rewritten in terms of finite difference operations $\{\hat{\partial}^{E}_{x}, \hat{\partial}^{E}_{y}, \hat{\partial}^{H}_{x}, \hat{\partial}^{H}_{y}\}$, defined as

$$\hat{\partial}_x^E E_{(i,j)} = \frac{E_{(i+1,j)} - E_{(i,j)}}{\Delta x} \tag{2.75}$$

$$\hat{\partial}_y^E E_{(i,j)} = \frac{E_{(i,j+1)} - E_{(i,j)}}{\Delta y} \tag{2.76}$$

$$\hat{\partial}_x^H H_{(i+\frac{1}{2},j+\frac{1}{2})} = \frac{H_{(i+\frac{1}{2},j+\frac{1}{2})} - H_{(i-\frac{1}{2},j+\frac{1}{2})}}{\Delta x} \tag{2.77}$$

$$\hat{\partial}_y^H H_{(i+\frac{1}{2},j+\frac{1}{2})} = \frac{H_{(i+\frac{1}{2},j+\frac{1}{2})} - H_{(i+\frac{1}{2},j-\frac{1}{2})}}{\Delta y} \tag{2.78}$$

where $\hat{\partial}_{x,y}^E$ acts only on components of $\mathbf{E}$ and $\hat{\partial}_{x,y}^H$ acts only on components of $\mathbf{H}$. We see that $\hat{\partial}_{x,y}^E$ is a forwards difference operation, and $\hat{\partial}_{x,y}^H$ is a reverse difference operator. This reduces equations 2.69 - 2.74 to the following matrix equations

$$-j\omega\mu_0 \begin{bmatrix} \mu_x \mathbf{H_x} \\ \mu_y \mathbf{H_y} \\ \mu_z \mathbf{H_z} \end{bmatrix} = \begin{bmatrix} 0 & -j\beta\mathbf{I} & \hat{\partial}_{\mathbf{y}}^{\mathbf{E}} \\ -j\beta\mathbf{I} & 0 & -\hat{\partial}_{\mathbf{x}}^{\mathbf{E}} \\ -\hat{\partial}_{\mathbf{y}}^{\mathbf{E}} & \hat{\partial}_{\mathbf{x}}^{\mathbf{E}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{E_x} \\ \mathbf{E_y} \\ \mathbf{E_z} \end{bmatrix} \tag{2.79}$$

$$j\omega \begin{bmatrix} \mathbf{D_x} \\ \mathbf{D_y} \\ \mathbf{D_z} \end{bmatrix} = \begin{bmatrix} 0 & -j\beta\mathbf{I} & \hat{\partial}_{\mathbf{y}}^{\mathbf{H}} \\ -j\beta\mathbf{I} & 0 & -\hat{\partial}_{\mathbf{x}}^{\mathbf{H}} \\ -\hat{\partial}_{\mathbf{y}}^{\mathbf{H}} & \hat{\partial}_{\mathbf{x}}^{\mathbf{H}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{H_x} \\ \mathbf{H_y} \\ \mathbf{H_z} \end{bmatrix} \tag{2.80}$$

Now, for a general anisotropic dielectric medium, the relation between $\mathbf{E}$ and $\mathbf{D}$ can be expressed as

$$\mathbf{D_x} = \epsilon_{xx}\mathbf{E_x} + \epsilon_{xy}\mathbf{E_y} + \epsilon_{xz}\mathbf{E_z} \tag{2.81}$$

$$\mathbf{D_y} = \epsilon_{yx}\mathbf{E_x} + \epsilon_{yy}\mathbf{E_y} + \epsilon_{yz}\mathbf{E_z} \tag{2.82}$$

$$\mathbf{D_z} = \epsilon_{zx}\mathbf{E_x} + \epsilon_{zy}\mathbf{E_y} + \epsilon_{zz}\mathbf{E_z} \tag{2.83}$$

which are used to define the nine ϵ_{ij} elements of the permittivity tensor at a given point in space. Inserting these relations into equations 2.79 - 2.80 yields the following relation for a full tensor dialectic material,

$$-j\omega\mu_0\mu_x\mathbf{H_x} = j\beta\mathbf{E_y} + \hat{\partial}_y^E\mathbf{E_z} \tag{2.84}$$

$$-j\omega\mu_0\mu_y\mathbf{H_y} = -j\beta\mathbf{E_x} - \hat{\partial}_x^E\mathbf{E_z} \tag{2.85}$$

$$-j\omega\mu_0\mu_z\mathbf{H_z} = -\hat{\partial}_y^E\mathbf{E_x} + \hat{\partial}_x^E\mathbf{E_y} \tag{2.86}$$

$$j\omega(\epsilon_{xx}\mathbf{E_x} + \epsilon_{xy}\mathbf{E_y} + \epsilon_{xz}\mathbf{E_z}) = j\beta\mathbf{H_y} + \hat{\partial}_y^H\mathbf{H_z} \tag{2.87}$$

$$j\omega(\epsilon_{yx}\mathbf{E_x} + \epsilon_{yy}\mathbf{E_y} + \epsilon_{yz}\mathbf{E_z}) = -j\beta\mathbf{H_x} - \hat{\partial}_x^H\mathbf{H_z} \tag{2.88}$$

$$j\omega(\epsilon_{zx}\mathbf{E_x} + \epsilon_{zy}\mathbf{E_y} + \epsilon_{zz}\mathbf{E_z}) = -\hat{\partial}_y^H\mathbf{H_x} + \hat{\partial}_x^H\mathbf{H_y} \tag{2.89}$$

Note that in Equations 2.84 - 2.89 each $\epsilon_{ij}\mathbf{E_j}$ or $\mu_i\mathbf{H_j}$ is a column vector with each element representing the value of $\epsilon^{ij}E^j$ or μ^iH^j at a corresponding Yee-mesh field component at a particular point in the computational domain, namely

$$\epsilon_{ij}\mathbf{E_j} = \begin{bmatrix} \epsilon^{ij}E^j_{1,1} \\ \epsilon^{ij}E^j_{2,1} \\ \vdots \\ \epsilon^{ij}E^j_{n,1} \\ \epsilon^{ij}E^j_{1,2} \\ \epsilon^{ij}E^j_{2,2} \\ \vdots \\ \epsilon^{ij}E^j_{n,m} \end{bmatrix} \tag{2.90}$$

$$\mu_i\mathbf{H_j} = \begin{bmatrix} \mu^iH^j_{1,1} \\ \mu^iH^j_{2,1} \\ \vdots \\ \mu^iH^j_{n,1} \\ \mu^iH^j_{1,2} \\ \mu^iH^j_{2,2} \\ \vdots \\ \mu^iH^j_{n,m} \end{bmatrix} \tag{2.91}$$

We can also observe the forward and backward difference matrices, expressed in equations 2.75 - 2.78 can be similarly represented in matrix form as

$$\hat{\partial}_{\mathbf{x}}^{\mathbf{E}} = \frac{1}{\Delta x}\begin{bmatrix} -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 1 \\ & & & & & -1 \end{bmatrix} \tag{2.92}$$

$$\hat{\partial}_{\mathbf{y}}^{\mathbf{E}} = \frac{1}{\Delta y}\begin{bmatrix} -1 & & & 1 & & \\ & -1 & & & \ddots & \\ & & \ddots & & & 1 \\ & & & \ddots & & \\ & & & & -1 & \\ & & & & & -1 \end{bmatrix} \tag{2.93}$$

$$\hat{\partial}_{\mathbf{x}}^{\mathbf{H}} = \frac{1}{\Delta x}\begin{bmatrix} 1 & & & & & \\ -1 & 1 & & & & \\ & -1 & \ddots & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & -1 & 1 \end{bmatrix} \tag{2.94}$$

$$\hat{\partial}_{\mathbf{y}}^{\mathbf{H}} = \frac{1}{\Delta y}\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ -1 & & & \ddots & & \\ & \ddots & & & 1 & \\ & & -1 & & & 1 \end{bmatrix} \tag{2.95}$$

Moreover, referencing figure 2-7 we observe that near the grid point (i, j) the

values of $\epsilon_{xj}, \epsilon_{yj}$, and ϵ_{zj} correspond to the material properties at the grid points $(i+\frac{1}{2}, j), (i, j+\frac{1}{2})$, and (i, j), respectively, which are the locations of D_x, D_y, and D_z, respectively.

Now, we are in position to create our Eigenvalue equation. By substituting the expressions for $\mathbf{E_z}$ and $\mathbf{H_z}$ from Equations 2.86 and 2.89, into Equations 2.84, 2.85, 2.87, and 2.88, we obtain an Eigenvalue equation for all four transverse field components:

$$\beta \begin{bmatrix} \mathbf{E_x} \\ \mathbf{E_y} \\ \mathbf{H_x} \\ \mathbf{H_y} \end{bmatrix} = \begin{bmatrix} \mathbf{M_{11}} & \mathbf{M_{12}} & \mathbf{M_{13}} & \mathbf{M_{14}} \\ \mathbf{M_{21}} & \mathbf{M_{22}} & \mathbf{M_{23}} & \mathbf{M_{24}} \\ \mathbf{M_{31}} & \mathbf{M_{32}} & \mathbf{M_{33}} & \mathbf{M_{34}} \\ \mathbf{M_{41}} & \mathbf{M_{42}} & \mathbf{M_{43}} & \mathbf{M_{44}} \end{bmatrix} \begin{bmatrix} \mathbf{E_x} \\ \mathbf{E_y} \\ \mathbf{H_x} \\ \mathbf{H_y} \end{bmatrix} \tag{2.96}$$

where the sub matrices $M_{ij}(i, j = 1, 2, 3, 4)$ are defined as:

$$\mathbf{M_{11}} = -j\left(\frac{\epsilon_{zx}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{E}} \tag{2.97}$$

$$\mathbf{M_{12}} = -j\left(\frac{\epsilon_{zy}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{E}} \tag{2.98}$$

$$\mathbf{M_{13}} = -\left(\frac{1}{\omega\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{E}}\hat{\partial}_{\mathbf{y}}^{\mathbf{H}} \tag{2.99}$$

$$\mathbf{M_{14}} = \left(\frac{1}{\omega\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{E}}\hat{\partial}_{\mathbf{x}}^{\mathbf{H}} + \omega\mu_y\mathbf{I} \tag{2.100}$$

$$\mathbf{M_{21}} = -j\left(\frac{\epsilon_{zx}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{y}}^{\mathbf{E}} \tag{2.101}$$

$$\mathbf{M_{22}} = -j\left(\frac{\epsilon_{zy}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{y}}^{\mathbf{E}} \tag{2.102}$$

$$\mathbf{M_{23}} = -\left(\frac{1}{\omega\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{y}}^{\mathbf{E}}\hat{\partial}_{\mathbf{y}}^{\mathbf{H}} - \omega\mu_x\mathbf{I} \tag{2.103}$$

$$\mathbf{M_{24}} = \left(\frac{1}{\omega\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{y}}^{\mathbf{E}}\hat{\partial}_{\mathbf{x}}^{\mathbf{H}} \tag{2.104}$$

$$\mathbf{M_{31}} = -\omega\epsilon_{yx}\mathbf{I} + \left(\frac{1}{\omega\mu_z}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{H}}\hat{\partial}_{\mathbf{y}}^{\mathbf{E}} + \omega\frac{\epsilon_{yz}\epsilon_{zx}}{\epsilon_{zz}}\mathbf{I} \tag{2.105}$$

$$\mathbf{M_{32}} = -\omega\epsilon_{yy}\mathbf{I} - \left(\frac{1}{\omega\mu_z}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{H}}\hat{\partial}_{\mathbf{x}}^{\mathbf{E}} + \omega\frac{\epsilon_{yz}\epsilon_{zy}}{\epsilon_{zz}}\mathbf{I} \tag{2.106}$$

$$\mathbf{M_{33}} = -j\left(\frac{\epsilon_{yz}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{y}}^{\mathbf{H}} \tag{2.107}$$

$$\mathbf{M_{34}} = -j\left(\frac{\epsilon_{yz}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{H}} \tag{2.108}$$

$$\mathbf{M_{41}} = \omega\epsilon_{yy}\mathbf{I} + \left(\frac{1}{\omega\mu_z}\right)\hat{\partial}_{\mathbf{y}}^{\mathbf{H}}\hat{\partial}_{\mathbf{y}}^{\mathbf{E}} - \omega\frac{\epsilon_{xz}\epsilon_{zx}}{\epsilon_{zz}}\mathbf{I} \tag{2.109}$$

$$\mathbf{M_{42}} = \omega\epsilon_{xy}\mathbf{I} - \left(\frac{1}{\omega\mu_z}\right)\hat{\partial}_{\mathbf{y}}^{\mathbf{H}}\hat{\partial}_{\mathbf{x}}^{\mathbf{E}} - \omega\frac{\epsilon_{xz}\epsilon_{zy}}{\epsilon_{zz}}\mathbf{I} \tag{2.110}$$

$$\mathbf{M_{43}} = j\left(\frac{\epsilon_{xz}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{H}} \tag{2.111}$$

$$\mathbf{M_{44}} = -j\left(\frac{\epsilon_{xz}}{\epsilon_{zz}}\right)\hat{\partial}_{\mathbf{x}}^{\mathbf{H}} \tag{2.112}$$

$$\tag{2.113}$$

Now since we are dealing with a full tensor material, we must adopt a more general PML which can handle anisotropic media, we will use that proposed by Teixeira and

Chew [33] as the absorbing boundary conditions. Thus, in the PML regions that permittivity and permeability tensors are given by,

$$\hat{\epsilon}_{\mathbf{PML}} = \begin{bmatrix} \frac{s_y s_z}{s_x}\epsilon_{xx} & s_z\epsilon_{xy} & s_y\epsilon_{xz} \\ s_z\epsilon_{yx} & \frac{s_x s_z}{s_y}\epsilon_{yy} & s_x\epsilon_{yz} \\ s_y\epsilon_{zx} & s_x\epsilon_{zy} & \frac{s_y s_x}{s_z}\epsilon_{zz} \end{bmatrix} \tag{2.114}$$

$$\hat{\mu}_{\mathbf{PML}} = \mu_0 \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_x s_z}{s_y} & 0 \\ 0 & 0 & \frac{s_y s_x}{s_z} \end{bmatrix} \tag{2.115}$$

where s_x, s_y and s_z are the so called complex PML parameters, where each is defined as

$$s_j = 1 - i\alpha_j \tag{2.116}$$

for $j = x, y, z$. The α_j's are chosen heuristically to control the field attenuation in the PML regions, for our application we choose:

$$\alpha_j = \alpha_{j,max}\left(\frac{\rho}{d}\right)^2 \tag{2.117}$$

for $j = x, y$ ($\alpha_z = 0$), ρ is defined as the distance in the j^{th} direct from the start of the PML region, and $\alpha_{j,max}$ is calculated using an assumed reflectivity value from the PML layer.

Now that we have designed a tool to calculate the non-reciprocal phase shift, we simply insert our calculated expression for ϵ_g into the mode solver to determine the non-reciprocal phase shift, as shown in Figure 2-9. This will become increasingly more accurate by inserting an increasingly more accurate value for ϵ_g. The full Matlab of implementation of this full epsilon tensor mode solver is given in Appendix A.

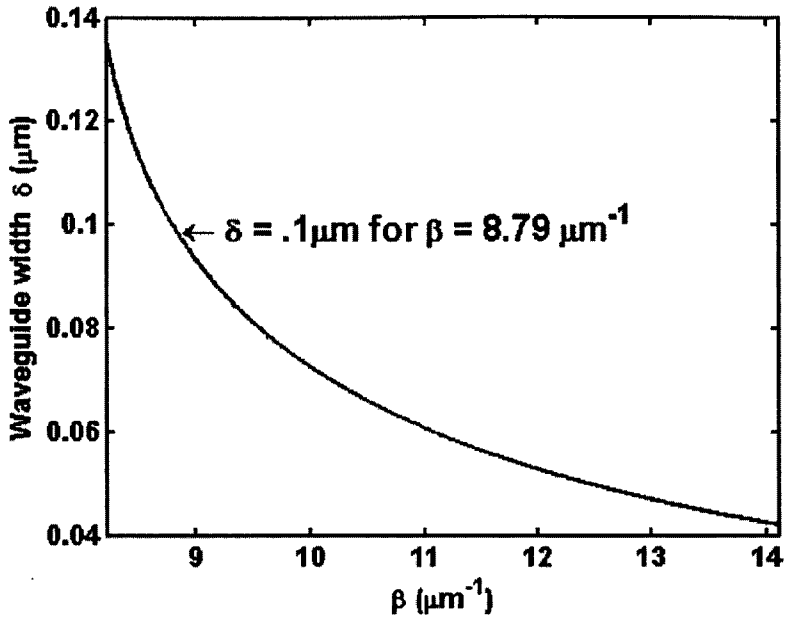


Figure 2-9: Nonreciprocal phase shift calculated using full permittivity tensor mode solver.

2.5 Integrated Isolator Designs

Now that we have determined the strength of the non-reciprocal phase shift, which as we recall is dependent upon our calculation of ϵ_g, we are now finally in a position to design an integrated optical isolator. In this section we will provide three device designs.

2.5.1 Passive Non-Reciprocal Mach-Zehnder Phase Shifter Isolator

The first design implements a Mach-Zehnder interferometer.

The nonreciprocal mach-zehnder isolator relies on the nonreciprocal phase shift between forward and backward propagating modes. Using this effect, a waveguide MachZehnder interferometer can be designed such that in the forward direction the modes in the two arms are in phase yielding constructive interference, whereas in the backward direction the modes are out of phase by 180° causing destructive interfer-

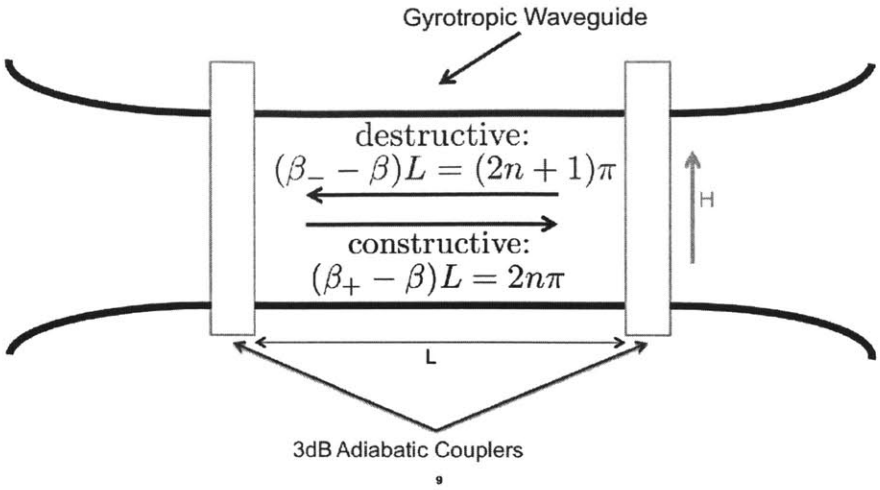


Figure 2-10: Diagram of the Mach-Zehnder isolator implementing nonreciprocal phase shift.

ence, this is diagrammatically shown in Figure 2-10.

For this passive Mach-Zehnder isolator the magnetic field is produced by an external magnetic, in either the equatorial or polar configurations as described in Figure 2-2.

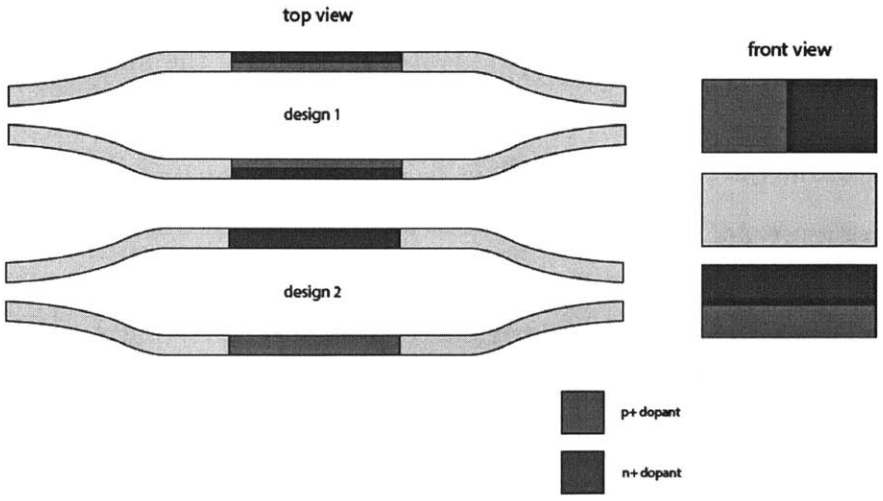


Figure 2-11: Two passive Mach-Zehnder isolator designs. Design 1 is in the Polar configuration, and design 2 is in the Equatorial configuration.

In Figure 2-11 the interferometer arms consist of nonreciprocal phase shift regions with opposite signs. They are in-plane magnetized perpendicular to mode propaga-

tion. As the magnetization is oppositely directed in the two arms, the nonreciprocal phase shift has an opposite sign so that the total length of the interferometer is only half of that of an interferometer with only one arm containing a nonreciprocal phase shift region.

In Figure 2-11 we see the use of P and N dopants, oppositely doped on either arm. In our application we also use Manganese dopants but doped only on one arm to create the non-reciprocal phase shift.

2.5.2 Non-reciprocal Ring Phase Shifter Isolator

The nonreciprocal ring resonator isolator works on a very similar similar principle as that of the Mach-Zehnder isolator. However, in the case of the ring we see a relative nonreciprocal phase shift depending on which direction the mode travels through the ring. We then, as in the case of the Mach-Zehnder isolator adjust the radius of the ring so that as light couples from the waveguide into the ring, we get constructive interferences, whereas in the opposite direction we get destructive interference.

As in the case of the Mach-Zehnder isolator we use an external magnetic to produce a magnetic field, and thus a gyrotropic media. In this case the magnet is located in the center of the ring, and the magnetic field is directed radially outward, so that at every point on the ring the magnetic field is in the equatorial configuration as described in Figure 2-2.

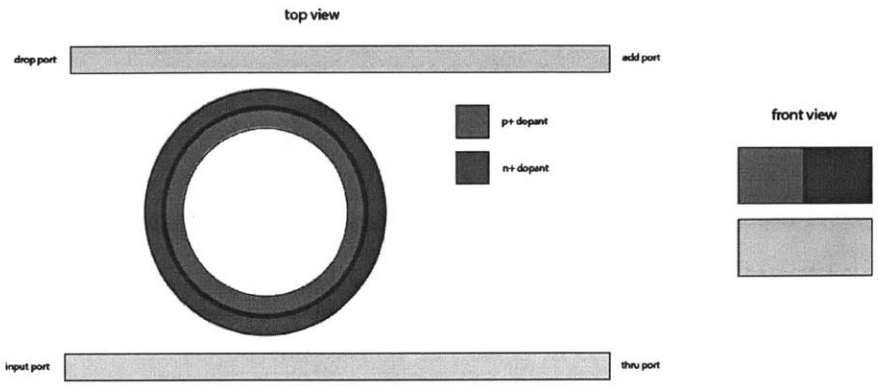


Figure 2-12: Microring Isolator in the polar configuration.

2.5.3 Active Non-reciprocal Mach-Zehnder Phase Shifter Isolator

This device is similar to the passive Mach-Zehnder isolator, except for the fact that the magnetic field does not come from an external off-chip magnet, but is instead generated on-chip through the use of metal coils, this is shown in Figure 2-13. This allows for a fully integrated solution, and since we can obtain a very dense and close magnetic field, higher performance.

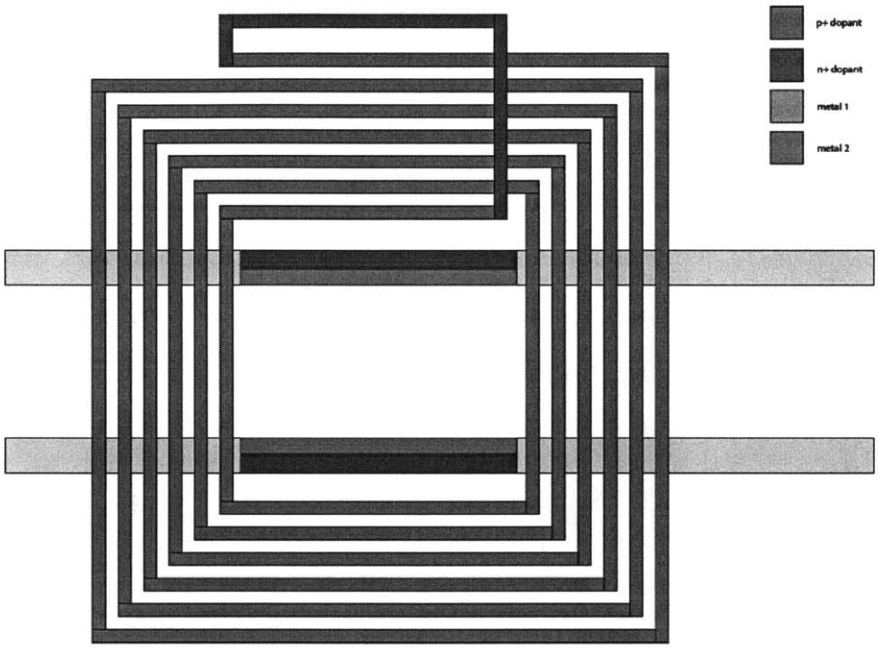


Figure 2-13: Active Mach-Zehnder isolator in the polar configuration. A magnetic field into or out of the page, is generated by the metal coils.

2.6 Conclusions

In conclusion, we present six designs for integrated optical isolators, by separating the design process into ϵ_g calculation, nonreciprocal phase shift calculation (analytical, perturbative, and numeric), and finally isolator design. We see that the nonreciprocal phase shift calculations agree within 20% between the three methods, which is surprising and counts in a positive way toward the validity and consistency of each

method, that can be applied to future device design.

2.7 Future Work

2.7.1 Experimental Determination of ϵ_g

Since, as stated earlier, the accuracy of the isolator design is intimately tied to the accuracy of the calculated ϵ_g, improving the accuracy of ϵ_g is paramount to the success of the device.

One such way to accomplish this is to get an estimate of the value of ϵ_g using experimental methods, which will give us another perspective on it's value and see how it compares to the derived quantity.

In this experiment we will measure the faraday rotation in a bulk Silicon wafer doped with Manganese. The experimental setup diagram is shown in Figure 2-15 and a picture of the actual experimental setup is shown in Figure 2-14.

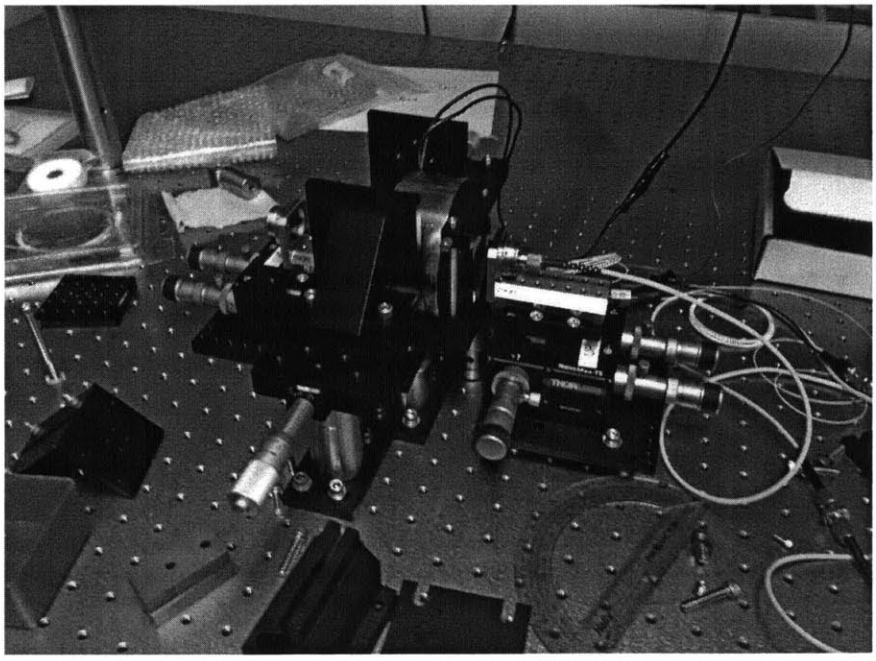


Figure 2-14: Image of experimental setup. This include the large 1.5 Tesla magnet in the center, the electrically driven fiber polarization modulator, the collimators, and the polarizer.

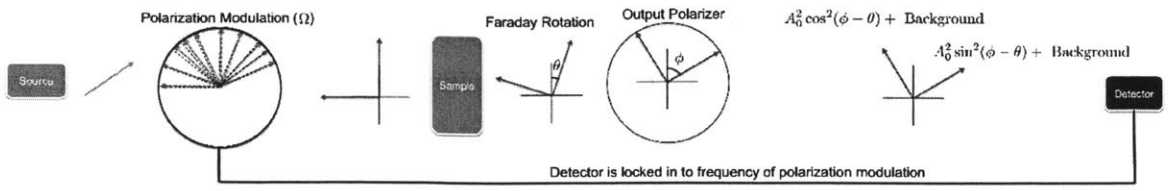


Figure 2-15: Diagram of the experimental setup shown above.

The reason for using Mn is that Mn-Doped Silicon creates an intrinsic magneto-optic material (rather than associated with the presence of ferromagnetic clusters or a foreign ferromagnetic phase).

The linearly polarized light incident on the sample is assumed to be propagating in the z direction and polarized along the x, y plane corresponding to the phase of the fiber polarization modulator, Ω. The electric eld of the light beam can be expressed in Jones matrix form as

$$E_0 = \begin{bmatrix} \cos(\Omega t) \\ \sin(\Omega t) \end{bmatrix} A_0 e^{-i\omega t + ikz} \tag{2.118}$$

where A_0 is the amplitude of the electric eld of the beam. Due to Faraday rotation the polarization of the light will be rotated by a small angle θ, which can be given by

$$E = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} A_0 e^{-i\omega t + ikz} \tag{2.119}$$

After the light travels through the analyzer, which is situated at an angle ϕ relative to the polarizer, we can represent the resultant field as

$$E = \begin{bmatrix} \cos(\phi-\theta)\cos(\phi) \\ \cos(\phi-\theta)\sin(\theta) \end{bmatrix} A_0 e^{-i\omega t + ikz} \tag{2.120}$$

the intensity measured at the detector is now given by

$$I = \cos^2(\phi-\theta)A_0^2 \tag{2.121}$$

We realize that maximum difference between the lowest and highest intensity (ΔI)

will happen at some optimal angle between the analyzer and the polarizer. Thus, by taking the first derivative of I with respect to θ, we see that

$$\frac{\partial I}{\partial \theta} = \sin(2\phi - 2\theta)A_0^2. \tag{2.122}$$

Because θ is small ($\theta \ll 1°$), it can easily be seen that a maximum ΔI is obtained when $\phi = 45°$, which means the intensity measured by the detector as expressed by Equation 2.121 is reduced to (subject to $\phi = 45°$)

$$I = \frac{1}{2}(1 + 2\theta)A_0^2 \tag{2.123}$$

Taking into consideration that the ac polarization modulization is sinusoidal, and because θ is proportional to B, the Faraday rotation angle can be written in the form $\theta = \theta_0 sin(\Omega t)$. Hence we see that Equation 2.123 can be simplified to

$$I = \frac{1}{2}(1 + 2\theta_0 \sin(\Omega t))A_0^2 = I_0 + \Delta I \sin(\Omega t). \tag{2.124}$$

By measuring the relative change of the light intensity $\frac{\Delta I}{I_0}$, the Faraday rotation angle can be determined by

$$\theta_0 = \frac{1}{2}\frac{2\theta_0 A_0^2}{A_0^2} = \frac{1}{2}\frac{\Delta I}{I_0}. \tag{2.125}$$

We can then easily obtain the Verdet coefficient from the follow relation

$$\theta_0 = V B_0 L, \tag{2.126}$$

where θ_0, L, B are all experimentally determined. Using these estimates I predict a realistic measurement of $\frac{\Delta I}{I_0}$ can be sensitive on the order on $\pm 10^{-5}$, which corresponds to a difference of the order 10^{-10} in the refractive index of left and right circularly polarized light in the medium using this technique. This implies a rotation of the plane of polarization of the order of 10^{-5} radians is measurable by this technique due to the elimination of broadband noise by the lock-in amplifier.

We then use this derived value of the Verdet constant to calculate an experimen-

tally obtained value for ϵ_g

Bibliography

[1] B. Saleh, *Fundamentals of Photonics*, ISBN-13: 978-0471358329, (2007)

[2] G. Budzyn, J. Rzepka, Back-reflection effects in a frequency-stabilized two-mode HeNelaser, Optics Communications, Volume 281, Issue 22, 15 November 2008, Pages 55925595

[3] Horst Dtsch, Norbert Bahlmann, Oleksandr Zhuromskyy, Manfred Hammer, Ludger Wilkens, Reinald Gerhardt, Peter Hertel, and Anatoly F. Popkov, "Applications of magneto-optical waveguides in integrated optics: review," J. Opt. Soc. Am. B 22, 240-253 (2005)

[4] O. Zhuromskyy, H. tsch, M. Lohmeyer, L. Wilkens, and P. Hertel, "Magnetooptical Waveguides with Polarization-Independent Nonreciprocal PhaseShift," J. Lightwave Technol. 19, 214- (2001)

[5] Ghosh S, Keyvavinia S, Van Roy W, Mizumoto T, Roelkens G, Baets R., Ce:YIG/Silicon-on-Insulator waveguide optical isolator realized by adhesive bonding, Opt Express. 2012 Jan 16;20(2):1839-48.

[6] J. B. Khurgin, Optical isolating action in surface plasmon polaritons, Appl. Phys. Lett. 89, 251115 (2006)

[7] J. B. Khurgin, Optical isolating action in surface plasmon polaritons, Appl. Phys. Lett. 89, 251115 (2006)

[8] F. Auracher, H. WITTE, "A New Design For an Integrated Optical Isolator", Opt. Comm., Volume 13, Number 4, December 1974

[9] Horst Do?tsch, Norbert Bahlmann, "Applications of magneto-optical waveguides in integrated optics: review", J. Opt. Soc. Am. B/Vol. 22, No. 1/January 2005

[10] A. B. Granovskia, Yu. P. Sukhorukov, "Ferromagnetism of Mn-Implanted Silicon: Magnetization and Magnetooptic Faraday Effect", ISSN 0021-3640, JETP Letters, 2007, Vol. 85, No. 7, pp. 335338. (2007)

[11] Aloke Jain, Jayant Kumar, Fumin Zhou, and Lian Li, "A simple experiment for determining Verdet constants using alternating current magnetic fields", Am. J. Phys. 67 (8), August (1999)

[12] G. Roelkens, D. Vermeulen, S. Selvaraja, R. Halir, W. Bogaerts and D. V. Thourhout, IEEE J. Selected Topics in Quantum Electronics, 17 3, 571-580, (2011)

[13] Intel Developer Forum 2003, url: "http://ic.tweakimg.net/ext/i/1109883395.png"

[14] M. Fan, M. A. Popovi?, and F. X. Krtner, OSA/CLEO CTuDD3 (2007).

[15] B. Wang, J. Jiang and G. P. Nordin, Photonics Tech. Let. 17, 9 (2005)

[16] K. Kintaka, Y. Kita, K. Shimizu, H. Matsuoka, S. Ura and J. Nishii, Opt. Let. 35, 12 (2010)

[17] G. Roelkens, D. V. Thourhout and R. Baets, Opt. Let. 32, 11 (2007)

[18] S. Scheerlinck, J. Schrauwen, F. V. Laere, D. Taillaert, D. V. Thourhout, and R. Baets, Opt. Exp. 15, 9625-9630 (2007)

[19] K. V. Acoleyen, H. Rogier, and Roel Baets, Opt. Exp. 18, 13655-13660 (2010)

[20] J. K. Doylend, M. J. R. Heck, J. T. Bovington, J. D. Peters, L. A. Coldren, and J. E. Bowers, Opt. Exp. 19, 21595-21604 (2011)

[21] Jie Sun , Erman Timurdogan , Ami Yaacobi, Ehsan shah Hosseini, Douglas Coolbaugh and Michael Watts, Nature (accepted for publication) (2012)

[22] A. Yaacobi, E. Timurdogan, and M. Watts, Opt. Lett. 37, pp. 1454-1456 (2012)

[23] A. Yaacobi and M. Watts, Opt. Lett. Early posting (2012)

[24] D. Dregely, R. T., J. Dorfmuller, R. Vogelgesang, K. Kern and H. Giessen, Nat. Commun. 2, 267 (2011)

[25] T. Kosako, Y. Kadoya, H. F. Hofmann, Nat. Photon. 4, 5 (2010), pp. 312 - 315

[26] J. Li and N. Engheta, IEEE 55, 11 (2007), pp. 3018-3026

[27] B. E. Little, S. T. Chu, H. A. Haus, J. Foresi and J. P. Laine, J. Lightwave Tec. 15, 6 (1997)

[28] H. A. Haus, chapter 7 in Waves and Fields in Optoelectronics, (Prentice-Hall, 1984), pp. 197-207

[29] H. A. Haus and W. P. Huang, IEEE 79, 10 (1991), pp. 1505-1518

[30] H. R. Philipp, SiO2 (Glass) and Si3N4 in Handbook of Opt. Const. of Solids, ed. E. D. Palik, (1998), pp. 760, 774

[31] Ming-yun Chen, Sen-ming Hsu, and Hung-chun Chang, " A nite-difference frequency-domain method for full-vectroial mode solutions of anisotropic optical waveguides with an arbitrary permittivity tensor",Vol. 17, No. 8 / OPTICS EXPRESS 5965, April 2009

[32] Thomas E. Murphy, "Design, Fabrication and Measurement of Integrated Bragg Grating Optical Filters" Thesis, MIT, February, 2001

[33] F. L. Teixeira and W. C. Chew, General closed-form PML constitutive tensors to match arbitrary bianisotropic and dispersive linear media, IEEE Microwave Guid. Wave Lett. 8, 223V-225 (1998)

[34] G. R. Hadley, High-accuracy nite-difference equations for dielectric waveguide analysis I: Uniform regions and dielectric interfaces, J. Lightwave Technol. 20, 12101218 (2002).

[35] G. R. Hadley, High-accuracy nite-difference equations for dielectric waveguide analysis II: Dielectric corners, J. Lightwave Technol. 20, 12191231 (2002).

[36] A. B. Fallahkhair, K. S. Li, and T. E. Murphy, Vector nite difference modesolver for anisotropic dielectric waveguides, J. Lightwave Technol. 26, 14231431 (2008)

[37] C. L. Xu, W. P. Huang, M. S. Stern, and S. K. Chaudhuri, Full-vectorial mode calculations by nite difference method, Proc. Inst. Electr. Eng. 141, 281286 (1994)

Appendix A

Full Permittivity Tensor Mode Solver Matlab Code

derivative_mesh.m: This function aligns the field and tensor components correctly on the Yee lattice

```
% create NSEWP
function [indooN,indooS,indooE,indooW,indooP,indooNW,indooNE,
    indooSW,indooSE...
      indoeN, indoeS,indoeE,indoeW,indoeP,indoeNW,indoeNE,
          indoeSW,indoeSE...
      indeoN,indeoS,indeoE,indeoW,indeoP,indeoNW,indeoNE,
          indeoSW,indeoSE...
      indeeN,indeeS,indeeE,indeeW,indeeP,indeeNW,indeeNE,
          indeeSW,indeeSE] = derivative_mesh(ind2,nx,ny)

% This function aligns the field and tensor components
    correctly on the Yee
% lattice
%
% USAGE:
```

```
11 %
12 %
13 %
14 % [indooN,indooS,indooE,indooW,indooP,indooNW,indooNE,indooSW
       ,indooSE...
15 %       indoeN,  indoeS,indoeE,indoeW,indoeP,indoeNW,indoeNE,
       indoeSW,indoeSE...
16 %       indeoN,indeoS,indeoE,indeoW,indeoP,indeoNW,indeoNE,
       indeoSW,indeoSE...
17 %       indeeN,indeeS,indeeE,indeeW,indeeP,indeeNW,indeeNE,
       indeeSW,indeeSE] = derivative_mesh(ind2,nx,ny)
18 %
19 %
20 % INPUT:
21 %
22 % nx,ny - size of index mesh
23 % ind2 - index mesh (doubled in size and interpolated
24
25 % OUTPUT:
26 %
27 % indices of correct gridding for derivative computations on
       Yee lattice
28 %
29 %
30 % AUTHOR:   Brad G Cordova (bcordova@mit.edu)
31 %           Version 1.0 (created July 2012)
32
33 indoo = ind2(1:2:end,1:2:end);
34 indoe = ind2(1:2:end,2:2:end);
35 indeo = ind2(2:2:end,1:2:end);
```

```
36 indee = ind2(2:2:end,2:2:end);
37
38 %% padding
39 indoo = [indoo(:,1),indoo,indoo(:,ny)];
40 indoo = [indoo(1,:);indoo;indoo(nx,:)];
41 indoe = [indoe(:,1),indoe,indoe(:,ny)];
42 indoe = [indoe(1,:);indoe;indoe(nx,:)];
43 indeo = [indeo(:,1),indeo,indeo(:,ny)];
44 indeo = [indeo(1,:);indeo;indeo(nx,:)];
45 indee = [indee(:,1),indee,indee(:,ny)];
46 indee = [indee(1,:);indee;indee(nx,:)];
47 size(indoo);
48 %%%%%%%
49 indooN = zeros(1,nx*ny);
50 indooS = zeros(1,nx*ny);
51 indooE = zeros(1,nx*ny);
52 indooW = zeros(1,nx*ny);
53 indooP = zeros(1,nx*ny);
54 indooNW = zeros(1,nx*ny);
55 indooNE = zeros(1,nx*ny);
56 indooSW = zeros(1,nx*ny);
57 indooSE = zeros(1,nx*ny);
58
59 indoeN = zeros(1,nx*ny);
60 indoeS = zeros(1,nx*ny);
61 indoeE = zeros(1,nx*ny);
62 indoeW = zeros(1,nx*ny);
63 indoeP = zeros(1,nx*ny);
64 indoeNW = zeros(1,nx*ny);
65 indoeNE = zeros(1,nx*ny);
```

```
66 indoeSW = zeros(1,nx*ny);
67 indoeSE = zeros(1,nx*ny);
68
69 indeoN = zeros(1,nx*ny);
70 indeoS = zeros(1,nx*ny);
71 indeoE = zeros(1,nx*ny);
72 indeoW = zeros(1,nx*ny);
73 indeoP = zeros(1,nx*ny);
74 indeoNW = zeros(1,nx*ny);
75 indeoNE = zeros(1,nx*ny);
76 indeoSW = zeros(1,nx*ny);
77 indeoSE = zeros(1,nx*ny);
78
79 indeeN = zeros(1,nx*ny);
80 indeeS = zeros(1,nx*ny);
81 indeeE = zeros(1,nx*ny);
82 indeeW = zeros(1,nx*ny);
83 indeeP = zeros(1,nx*ny);
84 indeeNW = zeros(1,nx*ny);
85 indeeNE = zeros(1,nx*ny);
86 indeeSW = zeros(1,nx*ny);
87 indeeSE = zeros(1,nx*ny);
88
89
90 %%%%%%%%
91 % NSEWP
92 indooN(:) = indoo(2:nx+1,3:ny+2);
93 indooS(:) = indoo(2:nx+1,1:ny);
94 indooE(:) = indoo(3:nx+2,2:ny+1);
95 indooW(:) = indoo(1:nx,   2:ny+1);
```

```
96  indooP(:) = indoo(2:nx+1,2:ny+1);
97  indooNW(:) = indoo(1:nx,   3:ny+2);
98  indooNE(:) = indoo(3:nx+2,   3:ny+2);
99  indooSW(:) = indoo(1:nx,1:ny);
100 indooSE(:) = indoo(3:nx+2,1:ny);
101
102 indoeN(:) = indoe(2:nx+1,3:ny+2);
103 indoeS(:) = indoe(2:nx+1,1:ny);
104 indoeE(:) = indoe(3:nx+2,2:ny+1);
105 indoeW(:) = indoe(1:nx,   2:ny+1);
106 indoeP(:) = indoe(2:nx+1,2:ny+1);
107 indoeNW(:) = indoe(1:nx,   3:ny+2);
108 indoeNE(:) = indoe(3:nx+2,   3:ny+2);
109 indoeSW(:) = indoe(1:nx,1:ny);
110 indoeSE(:) = indoe(3:nx+2,1:ny);
111
112 indeoN(:) = indeo(2:nx+1,3:ny+2);
113 indeoS(:) = indeo(2:nx+1,1:ny);
114 indeoE(:) = indeo(3:nx+2,2:ny+1);
115 indeoW(:) = indeo(1:nx,   2:ny+1);
116 indeoP(:) = indeo(2:nx+1,2:ny+1);
117 indeoNW(:) = indeo(1:nx,   3:ny+2);
118 indeoNE(:) = indeo(3:nx+2,   3:ny+2);
119 indeoSW(:) = indeo(1:nx,1:ny);
120 indeoSE(:) = indeo(3:nx+2,1:ny);
121
122 indeeN(:) = indee(2:nx+1,3:ny+2);
123 indeeS(:) = indee(2:nx+1,1:ny);
124 indeeE(:) = indee(3:nx+2,2:ny+1);
125 indeeW(:) = indee(1:nx,   2:ny+1);
```

```
126 indeeP(:) = indee(2:nx+1,2:ny+1);
127 indeeNW(:) = indee(1:nx,   3:ny+2);
128 indeeNE(:) = indee(3:nx+2,   3:ny+2);
129 indeeSW(:) = indee(1:nx,1:ny);
130 indeeSE(:) = indee(3:nx+2,1:ny);
```

index_mesh.m: This function aligns the field and tensor components correctly on the Yee lattice

```
1  function [dx dy nx ny ind] = index_mesh(dxy, width, height,
       epsilon, d_PML, PML_row, PML_column)
2  % This function creates an index mesh for the finite-
       difference
3  % mode solver.  The function will accommodate a generalized
       input matrix.
4  %
5  % USAGE:
6  %
7  %
8  % [dx dy nx ny ind] = index_mesh(dxy, width, height, epsilon)
9  % [dx dy nx ny ind] = index_mesh(dxy, width, height, epsilon,
       d_PML, PML_row, PML_column)
10 %
11 % INPUT
12 %
13 % dxy - vector [dx dy] horizontal and vertical grid spacing
14 % epsilon - structure of permitivities
15 % width - matrix defining horizontal widths of permittivity
       structure
16 % height - matrix defining vertical heights of permittivity
       structure
17 %
18 % OUTPUT
19 %
20 % dx,dy - horizontal and vertical grid spacing
21 % nx,ny - size of index mesh
22 % ind - index mesh
```

```
23 %
24 % AUTHOR:   Brad G Cordova (bcordova@mit.edu)
25 %           Version 1.0 (created July 2012)
26
27
28
29 dx = dxy(1);
30 dy = dxy(2);
31
32
33 numx = round(width/dx);
34 numy = round(height/dy);
35 nx = sum(numx);
36 ny = sum(numy);
37
38
39
40 AA = zeros(nx*ny,1);
41 kk = 0;
42
43 for ii = 1: size(numx,2)
44     for ss = 1: numx(ii)
45         for jj = 1: size(numy,2)
46             AA(kk+1: kk + numy(jj),1) = epsilon(jj,ii)*ones(
                   numy(jj),1);
47             kk = kk + numy(jj);
48         end
49     end
50 end
51 ind = AA;
```

```
52
53 if nargin==7 %Add PML to index mesh
54 nx_PML = round(d_PML/dx);
55 ny_PML = nx_PML;
56 ind = reshape(ind,ny,nx)';
57
58 triu_5_row =  triu(ones(nx_PML,ny_PML));
59 triu_5_col = triu(ones(nx_PML,ny_PML),1)';
60
61 triu_6_col =  fliplr(triu(ones(nx_PML,ny_PML),1));
62 triu_6_row = fliplr(triu(ones(nx_PML,ny_PML))');
63
64 triu_7_row =  fliplr(triu(ones(nx_PML,ny_PML)));
65 triu_7_col = fliplr(triu(ones(nx_PML,ny_PML),1)');
66
67 triu_8_col =  triu(ones(nx_PML,ny_PML),1);
68 triu_8_row = triu(ones(nx_PML,ny_PML))';
69
70 %PML boundary labels
71 %%%%%%%%%%%%%
72 % 5 | 3 | 6 %
73 % _ %%%%%% _ %
74 %   %     %  %
75 % 1 %     %2 %
76 % _ %     %_ %
77 %   %%%%%%  %
78 % 7 | 4 | 8 %
79 %%%%%%%%%%%%%
80
81 PML_1 = flipud(PML_row(:,ny_PML+1:ny_PML+ny,1));
```

```
82 PML_2 = PML_row(:,ny_PML+1:ny_PML+ny,2);
83 PML_3 = fliplr(PML_column(nx_PML+1:nx_PML+nx,:,3));
84 PML_4 = PML_column(nx_PML+1:nx_PML+nx,:,4);
85
86 PML_5 = flipud(PML_row(:,1:ny_PML,5)).*triu_5_row + fliplr(
       PML_column(1:nx_PML,:,5)).*triu_5_col;
87 PML_6 = PML_row(:,1:ny_PML,6).*triu_6_row + fliplr(
       PML_column(1:nx_PML,:,6)).*triu_6_col;
88 PML_7 = flipud(PML_row(:,1:ny_PML,7)).*triu_7_row +
       PML_column(1:nx_PML,:,7).*triu_7_col;
89 PML_8 = PML_row(:,1:ny_PML,8).*triu_8_row + PML_column(1:
       nx_PML,:,8).*triu_8_col;
90
91 PML_top = PML_3;
92 PML_bot = PML_4;
93 PML_left = [PML_5 PML_1 PML_7];
94 PML_right = [PML_6 PML_2 PML_8];
95
96 ind = [PML_top ind PML_bot];
97 ind = [PML_left; ind; PML_right];
98
99 % PML_row_top = [PML_row(:,1:ny_PML,5),PML_row(:,ny_PML+1:
       ny_PML+ny,3),PML_row(:,ny_PML+ny+1:ny_PML+ny+ny_PML,6)];
100 % PML_row_bottom = [PML_row(:,1:ny_PML,7),PML_row(:,ny_PML+1:
       ny_PML+ny,4),PML_row(:,ny_PML+ny+1:ny_PML+ny+ny_PML,8)];
101 % PML_column_left = PML_column(:,:,1);
102 % PML_column_right = PML_column(:,:,2);
103 %
104 %
105 %
```

```
106 % ind = [fliplr(PML_column_left) ind PML_column_right];
107 % ind = [flipud(PML_row_top); ind; (PML_row_bottom)];
108
109 [nx ny] = size(ind);
110
111 % figure;imagesc(abs(ind))
112
113 ind = reshape(ind',nx*ny,1);
114
115 end
```

eigen_modes.m: This function aligns the field and tensor components correctly on the Yee lattice

```
1  function [Ex Ey Ez Hx Hy Hz, neff, nx, ny] = eigen_modes(dxy,
       width, height, lambda, guess_index, n_modes, epsilon, mu, PML)
2  % This function computes all field components of a dielectric
       waveguide constructed from an arbitrary
3  % permitivity tensor, using the finite difference method.
4  %
5  % USAGE:
6  %
7  %
8  %
9  %[Ex Ey Ez Hx Hy Hz, neff, nx, ny] = eigen_modes(dxy, width,
       height, lambda, ...
10 %                                                guess_index,
       n_modes, epsilon, mu)
11 %
12 %[Ex Ey Ez Hx Hy Hz, neff, nx, ny] = eigen_modes(dxy, width,
       height, lambda, ...
13 %                                                guess_index,
       n_modes, epsilon, mu, PML)
14 %
15 %
16 % INPUT:
17 %
18 % lambda - optical wavelength
19 % guess_index - scalar shift to apply when calculating the
       eigenvalues.
20 %      This routine will return the eigenpairs which have an
21 %      effective index closest to this guess
```

```
22 % n_modes - the number of modes to calculate
23 % dxy - vector [dx dy] horizontal and vertical grid spacing
24 % epsilon - structure of permitivities
25 % mu - structure of permeabilites
26 %
27 % OUTPUT:
28 %
29 % Hx, Hy, Hz - calculated magnetic field components.
30 % Ex, Ey, Ez - calculated electric field components.
31 % nx,ny - size of index mesh
32 % neff - vector of modal effective indices
33 %
34 %
35 % AUTHOR:   Brad G Cordova (bcordova@mit.edu)
36 %           Version 1.0 (created July 2012)
37
38 dx = dxy(1); dy = dxy(2);
39 c_0 = 299792458;
40 omega = 2*pi*c_0/lambda;
41
42
43 if  nargin==9 %if PML
44
45 PML_row_epsilon = PML.row_epsilon;
46 PML_column_epsilon = PML.column_epsilon;
47 PML_row_mu = PML.row_mu;
48 PML_column_mu = PML.column_mu;
49 dPML = PML.width;
50
51 for ii = 1:9
```

```
52 [dx dy nx ny epsilon_mesh(:,ii)] = index_mesh(dxy, width,
       height, epsilon(:,:,ii),dPML,PML_row_epsilon,
       PML_column_epsilon);
53 epsilon_mesh2 = reshape(epsilon_mesh(:,ii),ny, nx)';
54 [X,Y] = meshgrid(1:ny, (1:nx)');
55 [MM,NN] = meshgrid(1:0.5:ny,(1:0.5:nx)');
56 epsilon_mesh2 = interp2(X, Y, epsilon_mesh2, MM, NN); %
       linear interpolation of the ind_mesh
57 [mx, my] = size(epsilon_mesh2);
58 epsilon_mesh2 = [epsilon_mesh2, epsilon_mesh2(:,my)]; % add
       column
59 epsilon_mesh2 = [epsilon_mesh2; epsilon_mesh2(mx,:)]; % add
       row
60 epsilon_mesh_double(:,:,ii) = epsilon_mesh2;
61 [mx, my] = size(epsilon_mesh_double(:,:,ii));
62 end
63
64 figure(100);
65 subplot(121)
66 imagesc(real(epsilon_mesh_double(:,:,1)')); title('Real Index
       Profile');
67 subplot(122)
68 imagesc(imag(epsilon_mesh_double(:,:,1)'));title('Imaginary
       Index Profile');
69
70 for ii = 1:9
71 [dx dy nx ny mu_mesh(:,ii)] = index_mesh(dxy, width, height,
       mu(:,:,ii),dPML,PML_row_mu,PML_column_mu);
72 mu_mesh2 = reshape(mu_mesh(:,ii),ny, nx)';
73 [X,Y] = meshgrid(1:ny, (1:nx)');
```

```
74 [MM,NN] = meshgrid(1:0.5:ny,(1:0.5:nx)');
75 mu_mesh2 = interp2(X, Y, mu_mesh2, MM, NN); % linear
       interpolation of the ind_mesh
76 [mx, my] = size(mu_mesh2);
77 mu_mesh2 = [mu_mesh2, mu_mesh2(:,my)]; % add column
78 mu_mesh2 = [mu_mesh2; mu_mesh2(mx,:)]; % add row
79 mu_mesh_double(:,:,ii) = mu_mesh2;
80 [mx, my] = size(mu_mesh_double(:,:,ii));
81 end
82
83 % figure; imagesc(mu_mesh_double(:,:,5));
84
85 else
86 for ii = 1:9
87 [dx dy nx ny epsilon_mesh(:,ii)] = index_mesh(dxy, width,
       height, epsilon(:,:,ii));
88 epsilon_mesh2 = reshape(epsilon_mesh(:,ii),ny, nx)';
89 [X,Y] = meshgrid(1:ny, (1:nx)');
90 [MM,NN] = meshgrid(1:0.5:ny,(1:0.5:nx)');
91 epsilon_mesh2 = interp2(X, Y, epsilon_mesh2, MM, NN); %
       linear interpolation of the ind_mesh
92 [mx, my] = size(epsilon_mesh2);
93 epsilon_mesh2 = [epsilon_mesh2, epsilon_mesh2(:,my)]; % add
       column
94 epsilon_mesh2 = [epsilon_mesh2; epsilon_mesh2(mx,:)]; % add
       row
95 epsilon_mesh_double(:,:,ii) = epsilon_mesh2;
96 [mx, my] = size(epsilon_mesh_double(:,:,ii));
97 end
98
```

```
99  figure(100); imagesc(real(epsilon_mesh_double(:,:,1)'));
100
101 for ii = 1:9
102 [dx dy nx ny mu_mesh(:,ii)] = index_mesh(dxy, width, height,
      mu(:,:,ii));
103 mu_mesh2 = reshape(mu_mesh(:,ii),ny, nx)';
104 [X,Y] = meshgrid(1:ny, (1:nx)');
105 [MM,NN] = meshgrid(1:0.5:ny,(1:0.5:nx)');
106 mu_mesh2 = interp2(X, Y, mu_mesh2, MM, NN); % linear
      interpolation of the ind_mesh
107 [mx, my] = size(mu_mesh2);
108 mu_mesh2 = [mu_mesh2, mu_mesh2(:,my)]; % add column
109 mu_mesh2 = [mu_mesh2; mu_mesh2(mx,:)]; % add row
110 mu_mesh_double(:,:,ii) = mu_mesh2;
111 [mx, my] = size(mu_mesh_double(:,:,ii));
112 end
113
114 % figure; imagesc(mu_mesh_double(:,:,5));
115 end
116
117
118
119 %%
120 fprintf(1,'Creating Index Mesh... \n');
121
122 [exx_oo_N,exx_oo_S,exx_oo_E,exx_oo_W,exx_oo_P,exx_oo_NW,
      exx_oo_NE,exx_oo_SW,exx_oo_SE...
123     exx_oe_N, exx_oe_S,exx_oe_E,exx_oe_W,exx_oe_P,exx_oe_NW,
          exx_oe_NE,exx_oe_SW,exx_oe_SE...
```

```
124     exx_eo_N,exx_eo_S,exx_eo_E,exx_eo_W,exx_eo_P,exx_eo_NW,
            exx_eo_NE,exx_eo_SW,exx_eo_SE...
125     exx_ee_N,exx_ee_S,exx_ee_E,exx_ee_W,exx_ee_P,exx_ee_NW,
            exx_ee_NE,exx_ee_SW,exx_ee_SE] =  derivative_mesh(
            epsilon_mesh_double(:,:,1),nx,ny);
126
127 [exy_oo_N,exy_oo_S,exy_oo_E,exy_oo_W,exy_oo_P,exy_oo_NW,
      exy_oo_NE,exy_oo_SW,exy_oo_SE...
128     exy_oe_N,  exy_oe_S,exy_oe_E,exy_oe_W,exy_oe_P,exy_oe_NW,
            exy_oe_NE,exy_oe_SW,exy_oe_SE...
129     exy_eo_N,exy_eo_S,exy_eo_E,exy_eo_W,exy_eo_P,exy_eo_NW,
            exy_eo_NE,exy_eo_SW,exy_eo_SE...
130     exy_ee_N,exy_ee_S,exy_ee_E,exy_ee_W,exy_ee_P,exy_ee_NW,
            exy_ee_NE,exy_ee_SW,exy_ee_SE] = derivative_mesh(
            epsilon_mesh_double(:,:,4),nx,ny);
131
132 [exz_oo_N,exz_oo_S,exz_oo_E,exz_oo_W,exz_oo_P,exz_oo_NW,
      exz_oo_NE,exz_oo_SW,exz_oo_SE...
133     exz_oe_N,  exz_oe_S,exz_oe_E,exz_oe_W,exz_oe_P,exz_oe_NW,
            exz_oe_NE,exz_oe_SW,exz_oe_SE...
134     exz_eo_N,exz_eo_S,exz_eo_E,exz_eo_W,exz_eo_P,exz_eo_NW,
            exz_eo_NE,exz_eo_SW,exz_eo_SE...
135     exz_ee_N,exz_ee_S,exz_ee_E,exz_ee_W,exz_ee_P,exz_ee_NW,
            exz_ee_NE,exz_ee_SW,exz_ee_SE] = derivative_mesh(
            epsilon_mesh_double(:,:,7),nx,ny);
136
137 [eyx_oo_N,eyx_oo_S,eyx_oo_E,eyx_oo_W,eyx_oo_P,eyx_oo_NW,
      eyx_oo_NE,eyx_oo_SW,eyx_oo_SE...
138     eyx_oe_N,  eyx_oe_S,eyx_oe_E,eyx_oe_W,eyx_oe_P,eyx_oe_NW,
            eyx_oe_NE,eyx_oe_SW,eyx_oe_SE...
```

```
139     eyx_eo_N, eyx_eo_S, eyx_eo_E, eyx_eo_W, eyx_eo_P, eyx_eo_NW,
            eyx_eo_NE, eyx_eo_SW, eyx_eo_SE...
140     eyx_ee_N, eyx_ee_S, eyx_ee_E, eyx_ee_W, eyx_ee_P, eyx_ee_NW,
            eyx_ee_NE, eyx_ee_SW, eyx_ee_SE] = derivative_mesh(
            epsilon_mesh_double(:,:,2), nx, ny);
141
142 [eyy_oo_N, eyy_oo_S, eyy_oo_E, eyy_oo_W, eyy_oo_P, eyy_oo_NW,
    eyy_oo_NE, eyy_oo_SW, eyy_oo_SE...
143     eyy_oe_N, eyy_oe_S, eyy_oe_E, eyy_oe_W, eyy_oe_P, eyy_oe_NW,
            eyy_oe_NE, eyy_oe_SW, eyy_oe_SE...
144     eyy_eo_N, eyy_eo_S, eyy_eo_E, eyy_eo_W, eyy_eo_P, eyy_eo_NW,
            eyy_eo_NE, eyy_eo_SW, eyy_eo_SE...
145     eyy_ee_N, eyy_ee_S, eyy_ee_E, eyy_ee_W, eyy_ee_P, eyy_ee_NW,
            eyy_ee_NE, eyy_ee_SW, eyy_ee_SE] = derivative_mesh(
            epsilon_mesh_double(:,:,5), nx, ny);
146
147 [eyz_oo_N, eyz_oo_S, eyz_oo_E, eyz_oo_W, eyz_oo_P, eyz_oo_NW,
    eyz_oo_NE, eyz_oo_SW, eyz_oo_SE...
148     eyz_oe_N, eyz_oe_S, eyz_oe_E, eyz_oe_W, eyz_oe_P, eyz_oe_NW,
            eyz_oe_NE, eyz_oe_SW, eyz_oe_SE...
149     eyz_eo_N, eyz_eo_S, eyz_eo_E, eyz_eo_W, eyz_eo_P, eyz_eo_NW,
            eyz_eo_NE, eyz_eo_SW, eyz_eo_SE...
150     eyz_ee_N, eyz_ee_S, eyz_ee_E, eyz_ee_W, eyz_ee_P, eyz_ee_NW,
            eyz_ee_NE, eyz_ee_SW, eyz_ee_SE] = derivative_mesh(
            epsilon_mesh_double(:,:,8), nx, ny);
151
152 [ezx_oo_N, ezx_oo_S, ezx_oo_E, ezx_oo_W, ezx_oo_P, ezx_oo_NW,
    ezx_oo_NE, ezx_oo_SW, ezx_oo_SE...
153     ezx_oe_N, ezx_oe_S, ezx_oe_E, ezx_oe_W, ezx_oe_P, ezx_oe_NW,
            ezx_oe_NE, ezx_oe_SW, ezx_oe_SE...
```

```
154       ezx_eo_N , ezx_eo_S , ezx_eo_E , ezx_eo_W , ezx_eo_P , ezx_eo_NW ,
              ezx_eo_NE , ezx_eo_SW , ezx_eo_SE ...
155       ezx_ee_N , ezx_ee_S , ezx_ee_E , ezx_ee_W , ezx_ee_P , ezx_ee_NW ,
              ezx_ee_NE , ezx_ee_SW , ezx_ee_SE ]  = derivative_mesh(
              epsilon_mesh_double(:,:,3) ,nx,ny);
156
157   [ezy_oo_N , ezy_oo_S , ezy_oo_E , ezy_oo_W , ezy_oo_P , ezy_oo_NW ,
        ezy_oo_NE , ezy_oo_SW , ezy_oo_SE ...
158       ezy_oe_N ,  ezy_oe_S , ezy_oe_E , ezy_oe_W , ezy_oe_P , ezy_oe_NW ,
              ezy_oe_NE , ezy_oe_SW , ezy_oe_SE ...
159       ezy_eo_N , ezy_eo_S , ezy_eo_E , ezy_eo_W , ezy_eo_P , ezy_eo_NW ,
              ezy_eo_NE , ezy_eo_SW , ezy_eo_SE ...
160       ezy_ee_N , ezy_ee_S , ezy_ee_E , ezy_ee_W , ezy_ee_P , ezy_ee_NW ,
              ezy_ee_NE , ezy_ee_SW , ezy_ee_SE ]  = derivative_mesh(
              epsilon_mesh_double(:,:,6) ,nx,ny);
161
162   [ezz_oo_N , ezz_oo_S , ezz_oo_E , ezz_oo_W , ezz_oo_P , ezz_oo_NW ,
        ezz_oo_NE , ezz_oo_SW , ezz_oo_SE ...
163       ezz_oe_N ,  ezz_oe_S , ezz_oe_E , ezz_oe_W , ezz_oe_P , ezz_oe_NW ,
              ezz_oe_NE , ezz_oe_SW , ezz_oe_SE ...
164       ezz_eo_N , ezz_eo_S , ezz_eo_E , ezz_eo_W , ezz_eo_P , ezz_eo_NW ,
              ezz_eo_NE , ezz_eo_SW , ezz_eo_SE ...
165       ezz_ee_N , ezz_ee_S , ezz_ee_E , ezz_ee_W , ezz_ee_P , ezz_ee_NW ,
              ezz_ee_NE , ezz_ee_SW , ezz_ee_SE ]  = derivative_mesh(
              epsilon_mesh_double(:,:,9) ,nx,ny);
166
167   [mu_x_oo_N , mu_x_oo_S , mu_x_oo_E , mu_x_oo_W , mu_x_oo_P , mu_x_oo_NW
        , mu_x_oo_NE , mu_x_oo_SW , mu_x_oo_SE ...
168       mu_x_oe_N ,  mu_x_oe_S , mu_x_oe_E , mu_x_oe_W , mu_x_oe_P ,
              mu_x_oe_NW , mu_x_oe_NE , mu_x_oe_SW , mu_x_oe_SE ...
```

```
169       mu_x_eo_N , mu_x_eo_S , mu_x_eo_E , mu_x_eo_W , mu_x_eo_P ,
              mu_x_eo_NW , mu_x_eo_NE , mu_x_eo_SW , mu_x_eo_SE ...
170       mu_x_ee_N , mu_x_ee_S , mu_x_ee_E , mu_x_ee_W , mu_x_ee_P ,
              mu_x_ee_NW , mu_x_ee_NE , mu_x_ee_SW , mu_x_ee_SE ] =
              derivative_mesh ( mu_mesh_double (: ,: ,1) ,nx , ny ) ;
171
172   [ mu_y_oo_N , mu_y_oo_S , mu_y_oo_E , mu_y_oo_W , mu_y_oo_P , mu_y_oo_NW
        , mu_y_oo_NE , mu_y_oo_SW , mu_y_oo_SE ...
173       mu_y_oe_N ,  mu_y_oe_S , mu_y_oe_E , mu_y_oe_W , mu_y_oe_P ,
              mu_y_oe_NW , mu_y_oe_NE , mu_y_oe_SW , mu_y_oe_SE ...
174       mu_y_eo_N , mu_y_eo_S , mu_y_eo_E , mu_y_eo_W , mu_y_eo_P ,
              mu_y_eo_NW , mu_y_eo_NE , mu_y_eo_SW , mu_y_eo_SE ...
175       mu_y_ee_N , mu_y_ee_S , mu_y_ee_E , mu_y_ee_W , mu_y_ee_P ,
              mu_y_ee_NW , mu_y_ee_NE , mu_y_ee_SW , mu_y_ee_SE ]  =
              derivative_mesh ( mu_mesh_double (: ,: ,5) ,nx , ny ) ;
176
177   [ mu_z_oo_N , mu_z_oo_S , mu_z_oo_E , mu_z_oo_W , mu_z_oo_P , mu_z_oo_NW
        , mu_z_oo_NE , mu_z_oo_SW , mu_z_oo_SE ...
178       mu_z_oe_N ,  mu_z_oe_S , mu_z_oe_E , mu_z_oe_W , mu_z_oe_P ,
              mu_z_oe_NW , mu_z_oe_NE , mu_z_oe_SW , mu_z_oe_SE ...
179       mu_z_eo_N , mu_z_eo_S , mu_z_eo_E , mu_z_eo_W , mu_z_eo_P ,
              mu_z_eo_NW , mu_z_eo_NE , mu_z_eo_SW , mu_z_eo_SE ...
180       mu_z_ee_N , mu_z_ee_S , mu_z_ee_E , mu_z_ee_W , mu_z_ee_P ,
              mu_z_ee_NW , mu_z_ee_NE , mu_z_ee_SW , mu_z_ee_SE ]  =
              derivative_mesh ( mu_mesh_double (: ,: ,9) ,nx , ny ) ;
181
182
183 %%
184
185 A11_P = 1j ./ dx .* ezx_oo_P ./ ezz_oo_P ;
```

```
186 A11_E = -1j./dx .* ezx_oo_E ./ ezz_oo_E;
187
188 A12_P = 1j./dx .* ezy_oo_P ./ ezz_oo_P;
189 A12_E = -1j./dx .* ezy_oo_E ./ ezz_oo_E;
190
191 A13_P = 1./(omega.*ezz_oo_P) ./ (dx.*dy);
192 A13_S = -1./(omega.*ezz_oo_P) ./ (dx.*dy);
193 A13_SE = 1./(omega.*ezz_oo_E) ./ (dx.*dy);
194 A13_E = -1./(omega.*ezz_oo_E) ./ (dx.*dy);
195
196 A14_P = -(1 ./ (omega.*ezz_oo_E) + 1 ./ (omega.*ezz_oo_P))./
      (dx.*dx) + omega .* mu_y_eo_P;
197 A14_E = 1./(omega.*ezz_oo_E) ./ (dx.*dx);
198 A14_W = 1./(omega.*ezz_oo_P) ./ (dx.*dx);
199
200 A21_P = 1j./dy .* ezx_oo_P ./ ezz_oo_P;
201 A21_N = -1j./dy .* ezx_oo_N ./ ezz_oo_N;
202
203 A22_P = 1j./dy .* ezy_oo_P ./ ezz_oo_P;
204 A22_N = -1j./dy .* ezy_oo_N ./ ezz_oo_N;
205
206 A23_P = (1 ./ (omega .* ezz_oo_P) + 1 ./ (omega .* ezz_oo_N))
      ./ (dy.*dy) - omega .* mu_x_oe_P;
207 A23_N = -1./(omega.*ezz_oo_N) ./(dy.*dy);
208 A23_S = -1./(omega.*ezz_oo_P) ./(dy.*dy);
209
210 A24_P = -1./(omega.*ezz_oo_P) ./(dy.*dx);
211 A24_N = 1./(omega.*ezz_oo_N) ./(dy.*dx);
212 A24_W = 1./(omega.*ezz_oo_P) ./(dy.*dx);
213 A24_NW = -1./(omega.*ezz_oo_N) ./(dy.*dx);
```

```
214
215 A31_P = -omega .* eyx_oe_P - 1./(omega.*mu_z_ee_E)./(dx.*dy)
      + eyz_oe_P .* omega .* ezx_oo_P ./ ezz_oo_P;
216 A31_N = 1./(omega.*mu_z_ee_E) ./(dx.*dy);
217 A31_W = 1./(omega.*mu_z_ee_P) ./(dx.*dy);
218 A31_NW = -1./(omega.*mu_z_ee_P) ./(dx.*dy);
219
220 A32_P = -omega .* eyy_oe_P + (1./(omega.*mu_z_ee_P)+1./(omega
      .*mu_z_ee_W))./(dx.*dx) + eyz_oe_P .* omega .* ezy_oo_P ./
       ezz_oo_P;
221 A32_E = -1./(omega.*mu_z_ee_P) ./(dx.*dx);
222 A32_W = -1./(omega.*mu_z_ee_W) ./(dx.*dx);
223
224 A33_P = -1j./dy .* eyz_oe_P ./ ezz_oo_P;
225 A33_S = 1j./dy .* eyz_oe_S ./ ezz_oo_S;
226
227 A34_P = 1j./dx .* ezy_oo_P ./ ezz_oo_P;
228 A34_W = -1j./dx .* ezy_oo_W ./ ezz_oo_W;
229
230 A41_P = omega .* exx_eo_P - (1./(omega .* mu_z_ee_P)+1./(
      omega .* mu_z_ee_S))./(dy.*dy) - exz_eo_P .* omega .*
      ezx_oo_P ./ezz_oo_P;
231 A41_N = 1./(omega.*mu_z_ee_P) ./(dy.*dy);
232 A41_S = 1./(omega.*mu_z_ee_S) ./(dy.*dy);
233
234 A42_P = omega .* exy_eo_P + 1./(omega.*mu_z_ee_P)./(dx.*dy) -
       exz_eo_P .* omega .* ezy_oo_P ./ezz_oo_P;
235 A42_S = -1./(omega.*mu_z_ee_S) ./(dx.*dy);
236 A42_E = -1./(omega.*mu_z_ee_P) ./(dx.*dy);
237 A42_SE = 1./(omega.*mu_z_ee_S) ./(dx.*dy);
```

```
238
239 A43_P = 1j./dy .* exz_eo_P ./ ezz_oo_P;
240 A43_S = -1j./dy .* exz_eo_S ./ ezz_oo_S;
241
242 A44_P = -1j./dx .* exz_eo_P ./ ezz_oo_P;
243 A44_W = 1j./dx .* exz_eo_W ./ ezz_oo_W;
244
245 %%
246 ii = zeros(nx,ny);
247 ii(:) = 1:nx*ny;
248 iall = zeros(1,nx*ny);
249 is = zeros(1,nx*(ny-1));
250 in = zeros(1,nx*(ny-1));
251 ie = zeros(1,(nx-1)*ny);
252 iw = zeros(1,(nx-1)*ny);
253 inw= zeros(1,(nx-1)*(ny-1));
254 ine= zeros(1,(nx-1)*(ny-1));
255 isw= zeros(1,(nx-1)*(ny-1));
256 ise= zeros(1,(nx-1)*(ny-1));
257
258
259 iall(:) = ii;
260 is(:) = ii(1:nx, 1:ny-1);
261 in(:) = ii(1:nx, 2:ny);
262 iw(:) = ii(1:nx-1,1:ny);
263 ie(:) = ii(2:nx, 1:ny);
264 inw(:)= ii(1:nx-1,2:ny);
265 ine(:)= ii(2:nx,2:ny);
266 isw(:)= ii(1:nx-1,1:ny-1);
267 ise(:)= ii(2:nx, 1:ny-1);
```

```
268
269 A11 = sparse ([iall,iw], ...
270     [iall,ie], ...
271     [A11_P(iall),A11_E(iw)]);
272
273 A12 = sparse ([iall,iw], ...
274     [iall,ie], ...
275     [A12_P(iall),A12_E(iw)]);
276
277 A13 = sparse ([iall,iw,in,inw], ...
278         [iall,ie,is,ise], ...
279         [A13_P(iall),A13_E(iw),A13_S(in),A13_SE(inw)]);
280
281 A14 = sparse ([iall,iw,ie], ...
282         [iall,ie,iw,], ...
283         [A14_P(iall),A14_E(iw),A14_W(ie)]);
284
285 A21 = sparse ([iall,is], ...
286         [iall,in], ...
287         [A21_P(iall),A21_N(is)]);
288
289 A22 = sparse ([iall,is], ...
290         [iall,in], ...
291         [A22_P(iall),A22_N(is)]);
292
293 A23 = sparse ([iall,is,in], ...
294         [iall,in,is], ...
295         [A23_P(iall),A23_N(is),A23_S(in)]);
296
297 A24 = sparse ([iall,ie,is,ise], ...
```

```
298           [iall,iw,in,inw], ...
299           [A24_P(iall),A24_W(ie),A24_N(is),A24_NW(ise)]);
300
301 A31 = sparse ([iall,ie,is,ise], ...
302           [iall,iw,in,inw], ...
303           [A31_P(iall),A31_W(ie),A31_N(is),A31_NW(ise)]);
304
305 A32 = sparse ([iall,iw,ie], ...
306           [iall,ie,iw,], ...
307           [A32_P(iall),A32_E(iw),A32_W(ie)]);
308
309 A33 = sparse ([iall,in], ...
310           [iall,is], ...
311           [A33_P(iall),A33_S(in)]);
312
313 A34 = sparse ([iall,ie], ...
314       [iall,iw], ...
315       [A34_P(iall),A34_W(ie)]);
316
317 A41 = sparse ([iall,is,in], ...
318           [iall,in,is], ...
319           [A41_P(iall),A41_N(is),A41_S(in)]);
320
321 A42 = sparse ([iall,iw,in,inw], ...
322           [iall,ie,is,ise], ...
323           [A42_P(iall),A42_E(iw),A42_S(in),A42_SE(inw)]);
324
325 A43 = sparse ([iall,in], ...
326           [iall,is], ...
327           [A43_P(iall),A43_S(in)]);
```

```
328
329 A44 = sparse ([iall,ie], ...
330     [iall,iw], ...
331     [A44_P(iall),A44_W(ie)]);
332
333
334 A = sparse([A11 A12 A13 A14; A21 A22 A23 A24; A31 A32 A33 A34
    ; A41 A42 A43 A44]); %Eigenvalue Matrix Equation
335
336 %%
337 fprintf(1,'Solving Eigenvalue Equation for Transverse Fields
    ... \n')
338 guess = (2*pi*guess_index/lambda);
339 options.tol = 1e-8;
340 options.disp = 0;
341 tic
342 [v,d] = eigs(A, speye(size(A)), n_modes, guess, options);
343 toc
344 neff = lambda*diag(d)/(2*pi);
345 Beta = diag(d);
346
347 %%
348 for ii=1:n_modes
349
350 Ex_tmp(:,:,ii) = reshape((v(1:nx*ny,ii)), nx,ny);
351
352 Ey_tmp(:,:,ii) = reshape((v(nx*ny+1:2*nx*ny,ii)), nx,ny);
353
354 Hx_tmp(:,:,ii) = reshape((v(2*nx*ny+1:3*nx*ny,ii)), nx,ny);
355
```

```
356 Hy_tmp(:,:,ii) = reshape((v(3*nx*ny+1:4*nx*ny,ii)), nx,ny);
357
358 end
359
360 fprintf(1,'Calculating Longitudinal Fields... \n \n')
361 [Ez_tmp,Hz_tmp] = longitudinal_fields(omega,dx,dy,mu_z_ee_P,
      ezx_oo_P,ezy_oo_P,ezz_oo_P,Ex_tmp,Ey_tmp,Hx_tmp,Hy_tmp);
362
363 for ii=1:n_modes
364 fprintf(1,'mode %d: n_eff = %e, beta = %2.12e \n',ii,neff(ii)
      ,Beta(ii))
365 end
366
367 %%
368 for ii=1:n_modes
369 Ex(:,:,ii) = Ex_tmp(:,:,ii);
370 Ey(:,:,ii) = Ey_tmp(:,:,ii);
371 Ez(:,:,ii) = Ez_tmp(:,:,ii);
372
373 Hx(:,:,ii) = Hx_tmp(:,:,ii);
374 Hy(:,:,ii) = Hy_tmp(:,:,ii);
375 Hz(:,:,ii) = Hz_tmp(:,:,ii);
376 end
```

longitudinal_fields.m: This function aligns the field and tensor components correctly on the Yee lattice

```
1  function [Ez,Hz] = longitudinal_fields(omega,dx,dy,mu_z_ee_P,
       ezx_oo_P,ezy_oo_P,ezz_oo_P,Ex,Ey,Hx,Hy)
2  % This function computes the longitudinal field components of
       a dielectric waveguide constructed from an arbitrary
3  % permitivity tensor, using the finite difference method.
4  %
5  % USAGE:
6  %
7  %
8  %
9  % [Ez,Hz] = longitudinal_fields(omega,dx,dy,mu_z_ee_P,
       ezx_oo_P,ezy_oo_P,ezz_oo_P,Ex,Ey,Hx,Hy)
10 %
11 %
12 % INPUT:
13 %
14 % Ex, Ey - calculated transverse electric field components
15 % Hx, Hy - calculated transverse magnetic field components
16 % omega - optical angular frequency
17 % dx - horizontal grid spacing (vector or scalar)
18 % dy - vertical grid spacing (vector or scalar)
19 % mu_z_ee_P - z-component of mu tensor
20 % ezx_oo_P - zx component of epsilon tensor
21 % ezy_oo_P - zy component of epsilon tensor
22 % ezz_oo_P - zz component of epsilon tensor
23 %
24 % OUTPUT:
25 %
```

```
26 % Hz - calculated longitudinal magnetic field.  This output
       will
27 %   have the same dimensions as Hx and Hy.
28 % Ez - calculated longitudinal magnetic field.  This output
       will
29 %   have the same dimensions as Ex and Ey.
30 %
31 %
32 % AUTHOR:   Brad G Cordova (bcordova@mit.edu)
33 %           Version 1.0 (created July 2012)
34
35
36 [nx,ny,nz] = size(Ex);
37
38 mu_z_ee_P = reshape(mu_z_ee_P,nx,ny);
39 ezx_oo_P = reshape(ezx_oo_P,nx,ny);
40 ezy_oo_P = reshape(ezy_oo_P,nx,ny);
41 ezz_oo_P = reshape(ezz_oo_P,nx,ny);
42
43 for ii=1:nz
44
45 Ex_j = pad_j(Ex(:,:,ii));
46 Hx_j = pad_j(Hx(:,:,ii));
47 Ey_i = pad_i(Ey(:,:,ii));
48 Hy_i = pad_i(Hy(:,:,ii));
49
50 Hz(:,:,ii) = -1./(1j*omega*mu_z_ee_P) .* ( 1/dx*(Ey_i(3:nx
       +2,:) - Ey_i(2:nx+1,:)) - 1/dy*(Ex_j(:,3:ny+2) - Ex_j(:,2:
       ny+1)) );
51
```

```
52 Ez(:,:,ii) = 1./(1j*omega*ezz_oo_P) .* (1/dx*(Hy_i(2:nx+1,:)
     - Hy_i(1:nx,:)) - 1/dy*(Hx_j(:,2:ny+1) - Hx_j(:,1:ny)) )
     ...
53                  - 1./ezz_oo_P .* (ezx_oo_P.*Ex(:,:,ii) +
                        ezy_oo_P.*Ey(:,:,ii));
54
55 end
```

pml_tensor.m: This function aligns the field and tensor components correctly on the Yee lattice

```
1  function [PML_row_epsilon PML_column_epsilon PML_row_mu
       PML_column_mu] = pml_tensor(dxy, width, height,
       pml_parameters, epsilon_tensor, mu_tensor)
2  % This function creates the PML boundary tensor using
       analytic function continuation into the complex plane.
3  %
4  % USAGE:
5  %
6  % [PML_row_epsilon PML_column_epsilon...
7  %      PML_row_mu PML_column_mu] = pml_tensor(dxy, width,
       height,...
8  %                                                pml_parameters,
       epsilon_tensor, mu_tensor)
9  %
10 % INPUT:
11 %
12 % epsilon_tensor - permitivity tensor
13 % mu_tensor - permeability tensor
14 % dxy - vector [dx dy] horizontal and vertical grid spacing
15 % width - matrix defining horizontal widths of permittivity
       structure
16 % height - matrix defining vertical heights of permittivity
       structure
17 %
18 % OUTPUT:
19 %
20 % PML_row_epsilon PML_column_epsilon - row and column of PML
       permittivity tensor
```

```
21 % PML_row_mu PML_column_mu - row and column of PML
       permeability tensor
22 %
23 % AUTHOR:   Brad G Cordova (bcordova@mit.edu)
24 %           Version 1.0 (created July 2012)
25
26
27 fprintf(1,'Creating PML Tensor... \n');
28
29 dx = dxy(1);
30 dy = dxy(2);
31
32
33 numx = round(width/dx);
34 numy = round(height/dy);
35 nx = sum(numx);
36 ny = sum(numy);
37
38
39 width_PML = pml_parameters.width;
40 height_PML = pml_parameters.height;
41 alpha_1_max_PML = pml_parameters.alpha_1_max;
42 alpha_2_max_PML = pml_parameters.alpha_2_max;
43 alpha_3_max_PML = pml_parameters.alpha_3_max;
44 alpha_4_max_PML = pml_parameters.alpha_4_max;
45
46 d_PML = width_PML;
47 nx_PML = round(width_PML/dx);
48 ny_PML = round(height_PML/dy);
49
```

```
50 for ii = 1:9
51 for jj=1:nx_PML
52     for qq=1:8
53 rho = jj*dx;
54 % alpha_x = alpha_x_max_PML.*(rho./d_PML).^2;
55 % alpha_y = alpha_y_max_PML.*(rho./d_PML).^2;
56
57 alpha_1 = alpha_1_max_PML*(rho./d_PML).^2;
58 alpha_2 = alpha_2_max_PML*(rho./d_PML).^2;
59 alpha_3 = alpha_3_max_PML*(rho./d_PML).^2;
60 alpha_4 = alpha_4_max_PML*(rho./d_PML).^2;
61
62 s1 = 1 - 1j*alpha_1;
63 s2 = 1 - 1j*alpha_2;
64 s3 = 1 - 1j*alpha_3;
65 s4 = 1 - 1j*alpha_4;
66
67 s_PML = [s1 1; s2 1; 1 s3; 1 s4; s1 s3; s2 s3; s1 s4; s2 s4];
68 sx = s_PML(qq,1);
69 sy = s_PML(qq,2);
70 sz = 1;
71
72 epsilon_PML_tensor = [(sy*sz/sx)*epsilon_tensor(1,1)  sz*
    epsilon_tensor(1,2)          sy*epsilon_tensor(1,3);
73                       sz*epsilon_tensor(2,1)          (sz*sx/
                            sy)*epsilon_tensor(2,2)  sx*
                            epsilon_tensor(2,3);
74                       sy*epsilon_tensor(3,1)          sx*
                            epsilon_tensor(3,2)          (sx*sy/
                            sz)*epsilon_tensor(3,3)];
```

```
75
76 PML_row_epsilon(jj,:,qq) = epsilon_PML_tensor(ii)*ones(1,ny
       +2*ny_PML);
77 PML_column_epsilon(:,jj,qq) = epsilon_PML_tensor(ii)*ones(nx
       +2*nx_PML,1);
78     end
79 end
80
81
82 for jj=1:nx_PML
83         for qq=1:8
84 rho = jj*dx;
85 % alpha_x = alpha_x_max_PML.*(rho./d_PML).^2;
86 % alpha_y = alpha_y_max_PML.*(rho./d_PML).^2;
87
88 alpha_1 = alpha_1_max_PML*(rho./d_PML).^2;
89 alpha_2 = alpha_2_max_PML*(rho./d_PML).^2;
90 alpha_3 = alpha_3_max_PML*(rho./d_PML).^2;
91 alpha_4 = alpha_4_max_PML*(rho./d_PML).^2;
92
93 s1 = 1 - 1j*alpha_1;
94 s2 = 1 - 1j*alpha_2;
95 s3 = 1 - 1j*alpha_3;
96 s4 = 1 - 1j*alpha_4;
97
98 s_PML = [s1 1; s2 1; 1 s3; 1 s4; s1 s3; s2 s3; s1 s4; s2 s4];
99 sx = s_PML(qq,1);
100 sy = s_PML(qq,2);
101 sz = 1;
102
```

```
103 mu_PML_tensor =  [ (sy*sz/sx)*mu_tensor(1,1)          0
                                                         0;
104                          0                                    (sz*sx/sy
                                )*mu_tensor(2,2)          0;
105                          0                                    0
                                                                  (sx*sy/
                                sz)*mu_tensor(3,3)];
106
107 PML_row_mu(jj,:,qq) = mu_PML_tensor(ii)*ones(1,ny+2*ny_PML);
108 PML_column_mu(:,jj,qq) = mu_PML_tensor(ii)*ones(nx+2*nx_PML
    ,1); %make bigger to account for expanded PML
109         end
110 end
111 end
```

mode_contour.m: This function aligns the field and tensor components correctly on the Yee lattice

```
1  function mode_contour(x,y,mode,dB,xyrange)
2  % Produces a contour plot (in dB) of one field component of
       the
3  % mode of an optical waveguide.
4  %
5  % USAGE:
6  %
7  % contourmode(x,y,mode);
8  % contourmode(x,y,mode,dBrange);
9  % contourmode(x,y,mode,dBrange,xyrange);
10 %
11 % INPUT:
12 %
13 % x,y - vectors describing horizontal and vertical grid
       points
14 % mode - the mode or field component to be plotted
15 % dBrange - contour levels to plot (in dB), with 0 dB
       corresponding
16 %   to the level |mode| = 1. default = (0:-3:-45)
17 % xyrange - axis range to use (optional)
18 %
19 % EXAMPLE: Make a contour plot of the magnetic field
       component Hx,
20 % with contours from 0 dB down to -50 dB, relative to the
       maximum
21 % value, in 5 dB increments.
22 %
23 %       contourmode(x,y,Hx/max(abs(Hx(:))),(0:-5:-50));
```

```
24 %
25 % NOTES:
26 %
27 % (1) This function uses the current color map to determine
       the
28 %     colors of each contour, with 0 dB corresponding to the
29 %     maximum color and -dbmax corresponding to the minimum
       color.
30 %     You can use the 'colormap' command to change the
       current
31 %     color map.
32 % (2) The aspect ratio of the plot box is automatically
       adjusted so
33 %     that the horizontal and vertical scales are equal.
34 % (3) The mode is not normalized or scaled in any way.
35 %
36 % AUTHOR:   Brad G Cordova (bcordova@mit.edu)
37 %           Version 1.0 (created July 2012)
38
39 x = real(x);
40 y = real(y);
41
42 if (nargin < 5)
43   xyrange = [min(x),max(x),min(y),max(y)];
44 end
45
46 if (size(mode) == [length(x)-1,length(y)-1])
47     x = (x(1:end-1) + x(2:end))/2;
48     y = (y(1:end-1) + y(2:end))/2;
49 end
```

```
50
51 if (nargin < 4) || isempty(dB)
52   dB = (0:-3:-45);
53 end
54
55 % Compute and plot contours
56 c = contourc(x,y,20*log10(abs(transpose(mode))),dB);
57 cmap = colormap;
58 ii = 1;
59 cla;
60 while (ii < length(c)),
61   level = c(1,ii);
62   n = c(2,ii);
63   jj = 1+round((length(cmap)-1)*(level - min(dB))/(max(dB)-
       min(dB)));
64   color = cmap(jj,:);
65   line(c(1,ii+1:ii+n),c(2,ii+1:ii+n),'Color',color);
66   ii = ii+n+1;
67 end
68
69 axis(xyrange);
70 set(gca,'PlotBoxAspectRatio',[xyrange(2)-xyrange(1) xyrange
     (4)-xyrange(3) 1],...
71         'Box','on');
```

mode_image.m: This function aligns the field and tensor components correctly on the Yee lattice

```
1  function [xf,yf,modebmp] = mode_image(x,y,mode,dx,dy)
2  % Produces a properly scaled color plot of a two-dimensional
3  % mode. This routine is especially useful when x and y are
4  % non-uniformly spaced vectors. In this case, the mode is
5  % interpolated over a uniformly-spaced grid before producing
6  % an image plot. The output can be directly saved to a file
7  % using the imwrite() function.
8  %
9  % USAGE:
10 %
11 % [xf,yf,modebmp] = imagemode(x,y,mode);
12 % [xf,yf,modebmp] = imagemode(x,y,mode,dx,dy);
13 %
14 % INPUT:
15 %
16 % x,y - vectors describing horizontal and vertical grid
       points
17 % mode - the mode or field component to be plotted
18 % dx, dy (optional) - fine grid spacing at which to
       oversample
19 %   (interpolate) the mode. If left unspecified, this
       routine
20 %   will use the smallest value of diff(x) and diff(y).
21 %
22 % OUTPUT:
23 %
24 % xf,yf - points at which the mode was interpolated
```

```
25 % modebmp - 8-bit unsigned integer array representing the
       mode
26 %      image
27 %
28 % AUTHOR:   Brad G Cordova (bcordova@mit.edu)
29 %           Version 1.0 (created July 2012)
30
31 x = real(x);
32 y = real(y);
33
34 if (size(mode) == [length(x)-1,length(y)-1])
35     x = (x(1:end-1) + x(2:end))/2;
36     y = (y(1:end-1) + y(2:end))/2;
37 end
38
39 if (nargin == 3)
40   [dx,ix] = min(diff(x));
41   [dy,iy] = min(diff(y));
42   xf = (min(x):dx:max(x))';
43   yf = (min(y):dy:max(y));
44   % line up with finest portion of grid
45   delta = dx*(interp1(xf,(1:length(xf)),x(ix+1)) - ...
46               round(interp1(xf,(1:length(xf)),x(ix+1))));
47   xf = xf + delta;
48   delta = dy*(interp1(yf,(1:length(yf)),y(iy+1)) - ...
49               round(interp1(yf,(1:length(yf)),y(iy+1))));
50   yf = yf + delta;
51   % eliminate points outside of range
52   kv = find((min(x) < xf) & (xf < max(x)));
53   xf = xf(kv);
```

```
54    kv = find((min(y) < yf) & (yf < max(y)));
55    yf = yf(kv);
56  else
57    xf = (min(x):dx:max(x))';
58    yf = (min(y):dy:max(y));
59  end
60
61  cmax = size(colormap,1)-1;
62
63  modebmp = uint8(transpose(interp2(y,x, ...
64                      abs(cmax*mode),yf,xf)));
65  image(xf,yf,flipud(modebmp));
66  set(gca,'YDir','normal');
67  v = [min(xf),max(xf),min(yf),max(yf)];
68  axis(v);
69  set(gca,'PlotBoxAspectRatio',[v(2)-v(1) v(4)-v(3) 1]);
```

pad_i.m: This function aligns the field and tensor components correctly on the Yee lattice

```
function B = pad_i(A)
%This function pads a matrix vertically
[nx,ny] = size(A);
B = [A(1,:);A;A(nx,:)];
end
```

pad_j.m: This function aligns the field and tensor components correctly on the Yee lattice

```
1 function B = pad_j(A)
2 %This function pads a matrix horizontally
3 [nx,ny] = size(A);
4 B = [A(:,1),A, A(:,ny)];
5 end
```

simple_wg.m: This function aligns the field and tensor components correctly on the Yee lattice

```
% Simple waveguide example for arbitrary permitivity tensor
    mode solver
%
% AUTHOR:   Brad G Cordova (bcordova@mit.edu)
%           Version 1.0 (created July 2012)
%
% All units are arbitrary but self consistant (and in this
    case all in SI)

clear all;
close all;
clc;
tic

%path where mode solver tools are located
addpath('./tools')

dx = 0.01e-6;
dy = dx;
dxy = [dx dy];

n_modes = 5;
guess_index = 2.42;

%basic parameters

lambda = 1550e-9; %free space wavelength
mu_0 = 4*pi*1e-7; %permeability of free space
```

```
27 epsilon_0 = 8.85418782e-12; % permittivity of free space
28 c_0 = 299792458; %speed of light in vacuum
29 omega = 2*pi*c_0/lambda; %angular frequency of free space
       light
30 k_0 = 2*pi/lambda; %free space wave vector
31
32 n_Si = 3.48; %index of refraction of Si
33 n_SiO = 1.445; %index of refraction of SiO2
34
35 %waveguide doping
36 doping_concentration = 1e26;%linspace(1e24,1e30,10)
37 alpha = 8.5e-18*doping_concentration/(1e6); %electrons
38 % alpha = 6.0e-18**doping_concentration/(1e6); %holes
39
40 alpha = alpha*100;
41 alpha_dB = alpha/100*10*log10(exp(1)); %dB cm-1
42
43 epsilon_Si = n_Si^2*epsilon_0;
44 epsilon_SiO = n_SiO^2*epsilon_0;
45 k_alpha = alpha*lambda/(4*pi);
46 n_Si = (3.48 + 1j*k_alpha);
47 epsilon_Si_complex = n_Si^2 * epsilon_0;
48
49 %Faraday Rotation
50 B0 = 1; % T
51 N = 5e24;
52 q = 1.60217646e-19;
53 m = 9.10938188e-31;
54 epsilon0 = 8.85418782e-12;
55
```

```
56 %1.541437e+00
57
58 u = 11.7e12; %
59 v = 32e-10; %
60
61
62
63 FR = B0*(u*lambda^2+v/lambda^2)*doping_concentration/(2.01e23
       );
64 epsilon_g = 3.48*FR*lambda/pi*epsilon_0;
65
66 % fprintf(1,'epsilon_g = %e \n\n',epsilon_g/epsilon_0);
67
68
69
70 %define index tensors here
71 epsilon_tensor_Si = [epsilon_Si  0           0;
72                      0           epsilon_Si  0;
73                      0           0           epsilon_Si];
74
75 epsilon_tensor_air = [epsilon_0  0           0;
76                       0          epsilon_0   0;
77                       0          0           epsilon_0];
78
79 epsilon_tensor_core = [epsilon_Si_complex        1j*epsilon_g
                         0;
80                        -1j*epsilon_g
                             epsilon_Si_complex            0;
81                         0                              0
```

```
                                epsilon_Si_complex];
82
83 epsilon_tensor_core2 = [epsilon_Si        0
                          1j*epsilon_g;
84                         0                 epsilon_Si
                              0;
85                         -1j*epsilon_g     0
                              epsilon_Si];
86
87
88 epsilon_tensor_SiO = [epsilon_SiO  0            0;
89                       0            epsilon_SiO  0;
90                       0            0            epsilon_SiO];
91 mu0_tensor =  [mu_0        0             0;
92                0           mu_0          0;
93                0           0             mu_0];
94
95
96
97
98
99 %define width and height of each segment
100 width = [1e-6, 0.48e-6, 1e-6];
101 height = [1e-6...
102           0.22e-6...
103           1e-6];
104
105 %epsilon structure
106 %here you create the structure corresponding to width and
      height grid variables
```

```
107 for ii = 1:9
108 epsilon(:,:,ii) = [epsilon_tensor_SiO(ii), epsilon_tensor_SiO
    (ii),epsilon_tensor_SiO(ii);
109                     epsilon_tensor_SiO(ii),
                           epsilon_tensor_core(ii),
                           epsilon_tensor_SiO(ii);
110                     epsilon_tensor_SiO(ii), epsilon_tensor_SiO
                           (ii),epsilon_tensor_SiO(ii)];
111
112                     %in this definition it goes xx,yx,zx,xy,
                           yy,zy,zx,xy,zz
113 end
114
115
116 %mu structure
117 %make sure mu and epsilon have the same number of matrix
    elements
118 %corresponding to width and height grid variables
119 for ii = 1:9
120 mu(:,:,ii) = [mu0_tensor(ii),mu0_tensor(ii), mu0_tensor(ii);
121               mu0_tensor(ii),mu0_tensor(ii), mu0_tensor(ii);
122               mu0_tensor(ii),mu0_tensor(ii), mu0_tensor(ii)];
123
124                 %in this definition it goes xx,yx,zx,xy,yy,zy
                       ,zx,xy,zz
125 end
126
127
128 %%
129 %Calculation of eigenmodes, and the output of fields
```

```
130 [Ex_tmp Ey_tmp Ez_tmp Hx_tmp Hy_tmp Hz_tmp, neff, nx, ny] =
      eigen_modes(dxy, width, height, lambda, guess_index, n_modes,
      epsilon, mu);
131
132
133 % beta_1 = 7.124457737634e+06;
134 % beta_2 = 7.118010554696e+06;
135 % nrps = abs(beta_1-beta_2)/(1e6)*180/pi %degrees per micron
136
137 %%
138 %Plotting routine
139 for ii=1:n_modes
140 figure
141
142
143 colormap(hot)
144
145 x = (1:nx)*dx;
146 xc = (1:nx)*dx - dx/2;
147 y = (1:ny)*dy;
148 yc = (1:ny)*dy - dy/2;
149
150 Ex = Ex_tmp(:,:,ii);
151 Ey = Ey_tmp(:,:,ii);
152 Ez = Ez_tmp(:,:,ii);
153
154 Hx = Hx_tmp(:,:,ii);
155 Hy = Hy_tmp(:,:,ii);
156 Hz = Hz_tmp(:,:,ii);
157
```

```
158
159 if max(max(abs(Hx))) > max(max(abs(Hy)))
160     hn = max(max(abs(Hx)));
161 else
162     hn = max(max(abs(Hy)));
163 end
164
165 if max(max(abs(Ex))) > max(max(abs(Ey)))
166     en = max(max(abs(Ex)));
167 else
168     en = max(max(abs(Ey)));
169 end
170
171
172 subplot(231)
173 mode_image(x,y,Ex/en);
174 title('E_x'); colorbar; axis xy; axis image;xlabel('x (m)');
      ylabel('y (m)');
175
176 subplot(232)
177 mode_image(x,y,Ey/en);
178 title('E_y'); colorbar; axis xy; axis image;xlabel('x (m)');
      ylabel('y (m)');
179
180 subplot(233)
181 mode_image(x,y,Ez/en);
182 title('E_z'); colorbar; axis xy; axis image;xlabel('x (m)');
      ylabel('y (m)');
183
184 subplot(234)
```

```
185 mode_image(x,y,Hx/hn);
186 title('H_x'); colorbar; axis xy; axis image;xlabel('x (m)');
      ylabel('y (m)');
187
188 subplot(235)
189 mode_image(x,y,Hy/hn);
190 title('H_y'); colorbar; axis xy; axis image;xlabel('x (m)');
      ylabel('y (m)');
191
192 subplot(236)
193 mode_image(x,y,Hz/hn);
194 title('H_z'); colorbar; axis xy; axis image;xlabel('x (m)');
      ylabel('y (m)');
195 end
196
197 for ii=1
198 figure
199 load('colormaps.mat')
200 colormap(whtrd)
201 set(gcf,'color','w');
202
203 x = (1:nx)*dx;
204 xc = (1:nx)*dx - dx/2;
205 y = (1:ny)*dy;
206 yc = (1:ny)*dy - dy/2;
207
208 Ex = Ex_tmp(:,:,ii);
209 Ey = Ey_tmp(:,:,ii);
210 Ez = Ez_tmp(:,:,ii);
211
```

```
212 Hx = Hx_tmp(:,:,ii);
213 Hy = Hy_tmp(:,:,ii);
214 Hz = Hz_tmp(:,:,ii);
215
216
217 if max(max(abs(Hx))) > max(max(abs(Hy)))
218     hn = max(max(abs(Hx)));
219 else
220     hn = max(max(abs(Hy)));
221 end
222
223 if max(max(abs(Ex))) > max(max(abs(Ey)))
224     en = max(max(abs(Ex)));
225 else
226     en = max(max(abs(Ey)));
227 end
228
229 mode_image(x,y,Ex/en);
230 title('E_x'); colorbar; axis xy; axis image;xlabel('x (m)');
    ylabel('y (m)');
231 end
```